Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель розподілу пулу завдань за обчислювальними ресурсами

(тема)

Виконав:

студент II курсу, групи СПзм-21-1

Корнієнко Д.Ю.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Філімончук Т.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Корнієнко Денису Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель розподілу пулу завдань за обчислювальними ресурсами

затверджена наказом по університету від " 24 " березня 2023 р. № 60 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 травня 2023 р.

3. Вхідні дані до роботи

1) імітаційне середовище моделювання GRASS;
2) модель розподілу завдань;
3) кортежі представлення завдань, ресурсів та методів розподілу.

4. Перелік питань, що потрібно опрацювати у роботі

1) показники, що характеризують обчислювальні системи;
2) методи та засоби розподілу обчислювальних ресурсів;
3) аналіз існуючих рішень для розподілу та балансування завдань;
4) аналіз існуючих моделей розподілу;
5) аналіз показників розподілу;
6) вбудовування модулю розподілу завдань до середовища моделювання GRASS.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд показників, що характеризують	27.03.23-01.04.23	
	обчислювальні системи		
2	Аналіз методів та засобів розподілу завдань	03.04.23-10.04.23	
	на обчислювальні ресурси кластерів		
3	Вибір інструментальних засобів	11.04.23-18.04.23	
4	Розробка моделі розподілу завдань	19.04.23-03.05.23	
5	Оформлення матеріалів кваліфікаційної роботи	04.05.23-08.05.23	
6	Подання кваліфікаційної роботи керівникові та		
	її попередній захист	10.05.23-11.05.23	
8	Подання кваліфікаційної роботи на рецензу-	12.05.23-16.05.23	
	вання		

Дата видачі завдання 27 березня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Філімончук Т.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 77 с., 15 рис., 10 табл., 1 дод., 33 джерела.

КЛАСТЕРНА ОБЧИСЛЮВАЛЬНА СИСТЕМА, ПЛАНУВАЛЬНИК, ВХІДНИЙ ПУЛ ЗАВДАНЬ, ОБЧИСЛЮВАЛЬНІ РЕСУРСИ, МЕТОДИ РОЗПОДІЛУ, МОДЕЛЮВАННЯ, ІМІТАЦІЙНЕ СЕРЕДОВИЩЕ МОДЕЛЮВАННЯ, ПЛАН РОЗПОДІЛУ.

Метою кваліфікаційної роботи є розробка моделі розподілу пулу завдань на обчислювальні ресурси кластеру.

У ході виконання кваліфікаційної роботи слід вирішити наступні задачі: проаналізувати існуючі на даний час методи розподілу пулу вхідних завдань в кластерних системах; розглянути існуючу математичну модель розподілу завдань на обчислювальні ресурси кластерних систем; розробити модель розподілу пулу вхідних завдань на обчислювальні ресурси кластерних систем з урахуванням відповідних критеріїв розподілу; провести ряд експериментів з розподілу пулу вхідних завдань в системі імітаційного моделювання GRASS.

Об'єктом дослідження кваліфікаційної роботи є процес розподілу пулу вхідних завдань на обчислювальні ресурси в кластерних системах. Предметом дослідження є методи розподілу та інформаційна технологія розподілу завдань в кластерних системах.

Методи досліджень засновані на використанні: теорії множин, загальної теорії систем, теорії імітаційного моделювання.

ABSTRACT

Master’s thesis: 77 pages, 15 figures, 10 tables, 1 appendice, 33 sources.

CLUSTER COMPUTING SYSTEM, SCHEDULER, INPUT TASK POOL, COMPUTING RESOURCES, DISTRIBUTION METHODS, SIMULATION, SIMULATION MODELING ENVIRONMENT, DISTRIBUTION PLAN.

The purpose of the qualification work is to develop a model for distributing the pool of tasks to the computing resources of the cluster.

In the course of the qualification work, the following tasks should be solved: analyze the currently existing methods of distributing the pool of incoming tasks in cluster systems; to consider the existing mathematical model of distribution of tasks on computing resources of cluster systems; to develop a model for the distribution of the pool of incoming tasks on the computing resources of cluster systems, taking into account the appropriate distribution criteria; conduct a series of experiments on the distribution of the pool of input tasks in the GRASS simulation modeling system.

The object of research of the qualification work is the process of allocating the pool of incoming tasks to computing resources in cluster systems. The subject of the research is distribution methods and information technology for the distribution of tasks in cluster systems.

Research methods are based on the use of: theory of sets, general theory of systems, theory of simulation modeling.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система

FCFS – алгоритм планування операційної системи, оснований на прямій черговості (англ., First-Come First-Served)

GRASS – імітаційне середовище моделювання (англ., GRid Advanced Simulation System)

HPF – алгоритм планування операційної системи з використанням пріоритетив (англ., Highest Priority First)

LIFO – алгоритм планування операційної системи, оснований на зворотній черговості (англ., Last In First Out)

ВСТУП

На даний час задача ефективного розподілу обчислювальної потужності обчислювальних ресурсів високопродуктивного середовища в процесі вирішення завдань кінцевих користувачів з метою покращення показників використання цих ресурсів залишається актуальною і характеризується високою обчислювальною складністю.

Вирішення сучасних інженерно-технічних задач потребує залучення значних обчислювальних ресурсів, а ресурсоємність процесу вирішення таких задач обумовлюється їх розмірністю, застосуванням оптимізаційних або перебірних методів розподілу, необхідністю проведення багатоваріантних розрахунків та іншими факторами. Прискорити процес вирішення таких задач можливо за допомогою розпаралелювання розрахунків та використання для їх проведення багатопроцесорних робочих станцій та серверів, які об'єднані у єдиний обчислювальний кластер.

Обчислювальний кластер – це складний програмно-апаратний комплекс, який вимагає спеціального програмного забезпечення, що надає користувачу ефективний план розподілу навантаження обчислювальної системи. Вибір вузлів обчислювального кластера, як правило, здійснюється за допомогою логіко-імовірнісного алгоритму багаторівневого конкретизуючого планування задач із заданими критеріями якості їх виконання. Процес планування, як правило, здійснюється у кілька етапів за допомогою агентів, які представляють собою вузли обчислювального кластера.

Процес планування розподілу включає:

- формування та облік множини доступних вузлів;

- подальша конкретизація сформованої множини шляхом виключення з неї перевантажених вузлів;

- побудова плану виконання завдання у вузлах.

1 СКЛАДОВІ ОЧИСЮВАЛЬНИХ СИСТЕМ

1.1 Показники, що характеризують обчислювальні системи

Першим найважливішим показником, що характеризує обчислювальні системи, є їхня продуктивність [1]. При формулюванні будь-якого завдання необхідно зіставляти продуктивність доступної обчислювальної системи з необхідними витратами часу для вирішення цього завдання або допустимою тривалістю очікування результатів (тривалістю відгуку). В якості істотного обмежуючого чинника виступає тривалість, протягом якої обчислювальні ресурси можуть бути задіяні на вирішення цього завдання, або реальний час, не більше якого доцільно отримати результати їхнього використання. Тому для всіх типів систем актуальною є проблема розподілу обмеженої продуктивності обчислювальної системи для вирішення функціональних завдань та застосування методів ефективної організації обчислювального процесу.

Другий найважливіший показник при використанні обчислювальних систем – це обсяг пам'яті для зберігання програм, констант та змінних. На затримку повідомлень перед обробкою значною мірою впливають тип та метод використання пам'яті для зберігання інформації. У сучасних обчислювальних системах обсяг зовнішньої пам'яті принципово може збільшуватися майже необмежено. Однак витрати продуктивності на пошук інформації у зовнішніх накопичувачах та на перепис її в оперативну пам'ять є суттєвими технічними обмеженнями для доцільного обсягу пам'яті, що використовується в обчислювальних системах.

Різноманітність видів пам'яті, що значно різняться за швидкодією, а також за вартістю зберігання одиниці інформації, призвело до застосування в обчислювальних системах багаторівневих систем пам'яті. Кожен наступний рівень пам'яті характеризується зниженням швидкодії та збільшенням можливого обсягу даних, що зберігаються. У даному напряму виникає задача

у визначенні обсягів пам'яті кожного рівня, що забезпечують зберігання заданого обсягу інформації за мінімального часу звернення до будь-яких даних та обмеженої вартості всієї пам'яті. Також слід розуміти, що характеристики пам'яті безпосередньо впливають на продуктивність обчислювальних систем і їх доводиться аналізувати спільно.

У загальному випадку в реальному часі продуктивність обчислювальних систем – це більш критичний параметр, ніж обсяг пам'яті.

1.2 Тривалість обробки заявок

Важливою складовою обчислювального процесу є тривалість виконання завдання [2]. На даний момент існує класифікація програм з тривалості обробки повідомлень.

Розглянемо перший тип програм. Для кожного певного завдання, якщо відсутні збої в апаратурі та всі дані, які було накопичено в пам'яті ЕОМ, існує певна послідовність виконання операцій ЕОМ, що призводить до результуючих значень. Отже, тривалість обробки таких даних можна визначити однозначно. У деяких програмах є один найімовірніший маршрут, який майже завжди реалізується за будь-яких вхідних даних. Тривалість обробки повідомлень, цим маршрутом може розглядатися як середня тривалість її виконання. При цьому ймовірність виконання програми іншими маршрутами вважається малою, а його дисперсія – близькою до нуля.

Другий тип програм, це коли програми містять тисячі умовних переходів та десятки циклів. Внаслідок цього, залежно від конкретних значень вихідних даних, утворюється дискретний спектр значень тривалостей обробки повідомлень деякого абонента. Такий спектр може містити багато тисяч значень і розглядатися як практично безперервний розподіл тривалостей. Програми цього типу характеризуються випадковими значеннями тривалостей виконання, різними залежно від змісту повідомлень та типів заявок. Передбачається, що наперед відомі розподілу тривалості

розв'язання задач або середні значення та деякі параметри розподілу. Вони дозволяють класифікувати надходження заявки за статистичними характеристиками тривалості реалізації в ЕОМ. Основним параметром такої класифікації є середній час обробки заявки і-го типу. При послідовному виконанні групи програм утворюється сукупність найімовірніших маршрутів, а множина практично незалежних випадкових чинників призводить до відхилення тривалостей реалізації щодо найімовірнішого часу реалізації програми.

При аналізі обслуговування (у більшості практичних випадків) можна задовольнитися основними характеристиками випадкових величин – їх першими двома моментами. При цьому значення середньої тривалості обслуговування, заявки і-го типу або інтенсивності обслуговування важливі не своїми абсолютними величинами, а в порівнянні з інтенсивністю того ж потоку і щодо тривалості обслуговування заявок інших типів. Програми другого типу характеризуються середньоквадратичним відхиленням тривалостей обслуговування, порівнянним або меншим середнього значення.

Експоненційний закон розподілу тривалостей обслуговування характерний тим, що більшість заявок обслуговується порівняно швидко. Для тривалості виконання реальних програм закон цей виконується рідко. Однак такий вид розподілу значно спрощує аналітичний розрахунок систем обслуговування та дає задовільні для практики результати, якщо аналіз упорядкування обмежується першими двома моментами часу очікування заявок. Експоненційний розподіл дозволяє отримати верхню оцінку тривалості очікування заявок або довжини черги при стаціонарному випадковому потоці заявок. Постійна тривалість обслуговування заявок і-го типу може використовуватись при аналізі впорядкування для отримання нижніх оцінок очікування та втрат при випадкових потоках заявок.

Третій тип програм, це коли внаслідок великої кількості неконтрольованих факторів іноді неможливо класифікувати тривалість реалізації заявок різних типів за деякими апріорними ознаками програм, що

викликаються. Інакше висловлюючись, дисперсія тривалості виконання заявок виявляється настільки велика, що їх класифікація не дозволяє віднести до кожного типу заявок середнє значення тривалості, яке досить відрізняється від тривалості виконання заявок інших типів. У цьому випадку передбачається відомою лише сумарна середня інтенсивність виконання всієї сукупності функціонуючих груп програм без класифікації їх за ознаками. Тому, інтегральний розподіл тривалостей обслуговування доводиться апроксимувати досить простим розподілом.

Найбільш зручним розподілом є експоненціальне, яке досить добре підтверджується експериментальними дослідженнями реальних тривалостей вирішення великих сукупностей завдань за відсутності селекції потоків.

1.3 Методи та засоби розподілу обчислювальних ресурсів

Методи та засоби розподілу [3-6] обчислювальних ресурсів та організації обчислювального процесу можна класифікувати за ступенем невизначеності апріорної інформації про основні динамічні характеристики надходження повідомлень та їх обслуговування.

Методи розподілу ресурсів першого типу характеризуються найбільш повною апріорною інформацією про моменти надходження даних, тривалість їх обробки, обсяг пам'яті, необхідний для зберігання вихідних повідомлень, програм та масивів даних, про зв'язки між програмами. У цьому випадку достовірна інформація дозволяє скласти план послідовності використання основних ресурсів обчислювальної системи на тривалий інтервал часу. Витрати на розподіл ресурсів є одноразовими і можуть бути здійснені поза оперативним функціонуванням обчислювальної системи.

У методах розподілу ресурсів другого типу використовуються статистичні характеристики процесу надходження повідомлень та ресурсів для їх реалізації, що дозволяють апріорно класифікувати повідомлення та викликані ними програми на кілька груп, що істотно відрізняються

статистичними параметрами. Кожну групу повідомлень можна описати набором параметрів та його статистичних розподілів. Вони досить повно характеризують динаміку надходження повідомлень і потребу ресурсів ВС для виконання програм, що їх викликають.

Методи розподілу ресурсів третього типу використовуються за відсутності параметра, що дозволяє апріорі класифікувати повідомлення та ресурси обчислювальної системи для їх реалізації. При цьому передбачаються відомими узагальнені середні характеристики всієї сукупності повідомлень, що надходять, і ресурсів обчислювальної системи, які необхідні для їх реалізації.

Відсутність параметрів, придатних для класифікації та ранжування програм, що викликаються, призводить до того, що ресурси розподіляються або зовсім випадково, або виділяються в міру надходження повідомлень з урахуванням оперативних оцінок потрібних ресурсів обчислювальної системи. Такі оцінки дозволяють перерозподіляти повідомлення, що надійшли, і, зокрема, реалізовувати в першу чергу ті, які вимагають мінімальних обсягів пам'яті або продуктивності обчислювальної системи.

1.3.1 Методи розподілу ресурсів у обчислювальних системах

Використання тих чи інших методів розподілу ресурсів залежить насамперед від динамічних характеристик використання обчислювальних систем та від ступеня зв'язку функціонування обчислювальної системи з реальним часом.

Статичні методи характеризуються можливістю великих витрат продуктивності та пам'яті обчислювальних систем на оптимізацію розподілу ресурсів, оскільки оптимізація проводиться одноразово на початок робочого функціонування системи [7]. При цьому передбачається, що апріорні дані про параметри, що враховуються при оптимізації, не змінюються протягом усього часу функціонування обчислювальної системи при проведеному

розподілі ресурсів. Статичний розподіл зводиться зазвичай до вирішення екстремальної задачі методами математичного програмування з відповідними обмеженнями. Допустимість великих витрат на розподіл та висока достовірність апріорної інформації дозволяють застосовувати точні методи оптимізації та отримувати розподіл ресурсів, що збігаються з оптимальними або дуже близькі до них. Подібні статичні завдання зустрічаються при розподілі обсягів багаторівневої пам'яті, при впорядкуванні послідовності розв'язання задач для однотипного регулярного надходження всіх заявок в один або кілька процесорів та деяких інших випадках.

Динамічні методи розподілу ресурсів реалізуються у процесі вирішення основних функціональних завдань даної обчислювальної системи. Для цього використовуються пам'ять та продуктивність тієї ж обчислювальної системи, яка потім застосовує результати розподілу. Тому допустимі витрати ресурсів обчислювальної системи на розподіл у разі дуже обмежені та не повинні перевищувати економії ресурсів, яка буде отримана, при використання відповідного методу. Чим достовірніша апріорна інформація, що використовується при розподілі, і чим довше вона зберігає своє значення, тим більш точним може бути розподіл ресурсів обчислювальної системи і тим довше можна користуватися результатами оптимізації. У тих випадках, коли апріорно відомі основні параметри заявок і їх обслуговування, а також та гарантовано, що протягом деякого часу будуть відсутні спотворення цих параметрів, розподіл ресурсів може проводитись одноразово на цей інтервал часу. Відповідно разові витрати на розподіл можуть бути підвищені та його доцільно проводити із застосуванням більш точних методів.

Для скорочення витрат на розподіл при частому його проведенні готуються правила, або дисципліни оперативної диспетчеризації, що забезпечують розподіли, досить близькі до оптимальних [8]. Ці правила ґрунтуються на попередніх дослідженнях різних методів розподілу ресурсів.

Численні технічні обмеження та невірність апріорної інформації призводять до доцільності застосування найпростіших правил та дисциплін, що наближено оптимізують розподіл ресурсів. Основою цих дисциплін є різні правила переваги чи пріоритетів.

1.3.1.1 Розподіл продуктивності обчислювальних систем за методами теорії розкладів

У ряді обчислювальних систем склад завдань, що розв'язуються, та тривалість їх вирішення вважатимуться відомими і не залежними від випадкових чинників. Це дозволяє розглядати розподіл ресурсів обчислювальної системи як строго детерміновану задачу, яка може бути сформульована в поняттях теорії розкладів [9] і вирішувати методами цієї теорії за таких припущень, коли:

- всі належні до виконання роботи повністю визначені та відомі, тобто відомі послідовність значень часу надходження заявок на їх виконання та тривалість вирішення кожної групи програм, що викликається;
- всі задані роботи мають бути повністю виконані, тобто відсутні втрати заявок через обмежену буферну пам'ять або інші фактори, зокрема через несправність машин;
- через відсутність проріджування потоку заявок при необмеженій буферної пам'яті обмежено сумарне завантаження;
- одна ЕОМ може одночасно вирішувати лише одне завдання і не допускається переривання до завершення реалізації заявки на завдання.

1.3.1.2 Методи оперативної диспетчеризації в однопроцесорних ЕОМ

Нижче основним показником якості розподілу ресурсів розглядається еквівалентна зміна продуктивності обчислювальної системи при різних дисциплінах порівняно з найпростішими еталонними дисциплінами

розподілу ресурсів. У всіх аналізованих дисциплінах за наявності переривання наступне обслуговування триває з перерваного стану, отже сумарна тривалість обслуговування не змінюється за будь-якої кількості переривань, а то й враховуються витрати продуктивності ЕОМ з їхнього виконання.

У безпріоритетних дисциплінах за відсутності апріорної інформації про характеристики заявок та їх обслуговування немає причин для переваги будь-яким з них. Порядок обслуговування у разі може бути випадковим чи враховувати послідовність надходження заявок. Найпростішими варіантами є обслуговування за принципом «першим прийшов – першим обслужений» або «останній прийшов – перший обслужений».

Розглянемо більш докладніше дисципліни з квантуванням часу обслуговування [1]. Іноді виникає ситуація, коли відсутня апріорна інформація про тривалість виконання програм. Тоді слід запускати у обчислювальних системах програми на виконання, які потребують мінімальний час використання обчислювальних ресурсів системи. Для цього застосовується динамічна адаптація у процесі виконання таких програм. Для цього в системі виділяються деякі інтервали (кванти) часу, у яких завдання вирішується без переривання. Якщо завдання не вирішено повністю в межах такого кванта, то воно переривається, повертається в чергу і на виконання пропускаються інші програми. Отже, обчислювальні ресурси оперативно надаються переважно коротким за тривалістю реалізації завданням, а довгі вирішуються протягом кількох квантів, послідовно пропускаючи коротші. Дисципліни очікування та продовження обслуговування після виділення чергового кванта часу на виконання розрізняються кількістю черг для очікування та методами зміни розміру квантів залежно від тривалості очікування або кількості вже використаних квантів.

Якщо характеристики обслуговування заявок різні (залежно від їх типу чи джерела), то з'являється можливість переваги при обслуговуванні деяких із них. Найбільш чітко особливості пріоритетного обслуговування

виявляються при аналізі дисциплін з відносними та абсолютними пріоритетами. Ці дисципліни можливо розрізняти за використанням відомостей про характеристики обслуговування заявок, які надійшли до системи та заявки, яка на даний час обслуговується. Такими даними є середня тривалість обслуговування заявки відповідного типу та штраф за очікування у черзі.

Дисципліна з відносними пріоритетами [2] характеризується тим, що заявки впорядковуються та обслуговуються в залежності від середньої тривалості обслуговування та штрафу за очікування у черзі. Основним показником, що використовується для розподілу типів заявок за пріоритетними рівнями, є відношення штрафу за одиницю часу очікування у черзі до середньої тривалості обслуговування заявок даного типу. При цій дисципліні дисперсія та інші характеристики тривалостей обслуговування ефективності дисципліни не впливають і обслуговування будь-якої заявки не переривається.

Дисципліна з абсолютними пріоритетами [2] характеризується перериванням обслуговування будь-якої заявки у разі виникнення заявки вищого пріоритету. У подальшому перервану заявку буде продовжено після завершення обслуговування заявки вищого пріоритету. Розподіл типів заявок за пріоритетними рівнями, так само як і в попередньому випадку, є апріорним відповідно до значень відносини, однак у цьому випадку на ефективність дисципліни, впливає дисперсія тривалостей обслуговування.

1.4 Критерії ефективності розподілу ресурсів у обчислювальних системах

Реальні обмеження на продуктивність та інші характеристики обчислювальних систем призводять до очікування результатів або до неповного вирішення завдань. Нерівноцінність завдань за допустимим часом затримки або допустимої ймовірності пропуску рішення, а також відмінності

параметрів обчислювальних систем дозволяють змінювати якість розв'язання задач виділенням відповідних ресурсів системи. Упорядкування послідовності розв'язання завдань та раціональне використання ресурсів обчислювальної системи скорочує запізнення у розв'язанні задач і певною мірою еквівалентно підвищенню продуктивності обчислювальної системи. Продуктивність обчислювальної системи на деякому наборі завдань – це найважливіші критерії ефективності системи загалом і методів розподілу ресурсів зокрема [1,2]. Більш складні та ефективні методи розподілу ресурсів, природно, вимагають великих ресурсів тієї ж обчислювальної системи для реалізації. Таким чином, виникає задача визначення оптимальних параметрів обчислювальної системи для вирішення деякої сукупності задач з урахуванням якості розподілу ресурсів та витрат на їх реалізацію.

Наприклад, слід розглянути критерії ефективності розподілу ресурсів з використанням штрафів. Очікування результатів і пропуск вирішення завдань можна описати деякими втратами ефективності або штрафами за те, що завдання не вирішуються або вирішуються не своєчасно. Залежно від призначення та типу обчислювальної системи затримка може оцінюватись або за середньою тривалістю очікування результатів, або за величиною перевищення фіксованого (припустимого) часу очікування. Відповідно передбачається, що повідомлення або заявка знецінюються або пропорційно до тривалості затримки, або стрибком при перевищенні допустимого (директивного) часу очікування.

Таким чином, підводячи підсумок попереднього аналізу, для того, щоб оцінити ефективність функціонування моделі обчислювальної системи, необхідно розробити адекватну методику оцінки результатів її роботи. Вона має виражатися у виявленні певних кількісних та якісних критеріїв оцінки ефективності виконання різноманітних завдань у межах функціональної моделі, що використовується.

Критерій ефективності – це правило, що служить для порівняльної

оцінки якості варіантів обчислювальних систем. Будуються такі критерії ефективності з урахуванням приватних характеристик ефективності [10]. В якості критеріїв моно розглядати наступні характеристики:

- загальна продуктивність обчислювальної системи, що була змодельована;
- час виконання одного й того ж набору завдань щодо різних алгоритмів розподілу;
- рівень використання обчислювальної інфраструктури протягом усього часу дослідження;
- відношення обчислювальних вузлів, які задіяні у виконанні завдань та тих, що простоюють;
- відношення завдань, які на даний час виконуються та тих, що очікують виконання.

Сучасні системи моделювання обчислювальних систем дозволяють досить точно оцінити всі вищезгадані характеристики як за допомогою журналів логів, що містять подібну інформацію, так і за допомогою засобів графічного представлення вихідних даних.

1.5 Класифікація стратегій розподілу навантаження

Стратегія розподілу інформаційних та обчислювальних ресурсів – це комплекс управлінських рішень, які використовують низку правил, що використовують інформацію про тип обчислювальної системи, її структуру та масштаби, а також орієнтована на контроль та управління структурними елементами даної системи [11].

На даний час існує безліч стратегій розподілу, з яких можна сформувати таку класифікацію:

- за способом обліку динаміки;
- за принципом управління;
- за ознакою універсальності;

- з урахуванням продуктивності вузлів;

- з використання знань про вільні обчислювальні ресурси системи.

За способом обліку динаміки можна виділити статичні, напівдинамічні та динамічні стратегії розподілу. Статична стратегія, зазвичай, передбачає певний (заздалегідь) план розподілу обчислювальних ресурсів системи. При використанні напівдинамічної стратегії розподілу план розподілу задається на початок розрахунків з урахуванням деяких даних стосовно структури обчислювальної системи. Використання динамічної стратегії розподілу обчислювальних ресурсів передбачає періодичну зміну плану в залежності від ряду параметрів, які було перелічено заздалегідь. Наведені стратегії дозволяють здійснювати балансування навантаження обчислювальної системи: наприклад, при виявленні нових обчислювальних вузлів або відмови вже працюючих.

За принципом управління стратегії розподілу ресурсів можна розділити на стратегії з децентралізованим, централізованим та ієрархічним управлінням. При використанні централізованих стратегій розподілу передбачається наявність центрального елемента – планувальника, який здійснює розподіл ресурсів з урахуванням інформації від кожного окремо взятого обчислювального вузла розподіленої системи. Децентралізовані алгоритми розподілу використовуються на кожному обчислювальному вузлі незалежно або з урахуванням даних, які було отримано від найближчих обчислювальних вузлів. Ієрархічний підхід при розподілі, у свою чергу, поєднує властивості централізованої та децентралізованої стратегії. При використанні ієрархічного підходу головний планувальник (метапланувальник) розподіляє завдання, що надходять на вхід обчислювальної системи, у відповідності з визначеним раніше набором правил, а глобальна черга завдань розбивається на кілька підчерг. Кількість підчерг регулюється кількістю елементів у кожному рівні ієрархії.

За ознакою універсальності стратегії розподілу можна поділити на універсальні та спеціалізовані стратегії. Спеціалізовані стратегії зазвичай

орієнтуються на специфічність конкретної розподіленої системи. На відміну від них, універсальні стратегії використовуються для різних обчислювальних систем, і орієнтовані на вирішення безлічі обчислювальних завдань.

За використанням механізмів прогнозу стратегії розподілу можна поділити на прогностичні стратегії та стратегії, які при розподілі не використовують механізми прогнозів. Стратегії, які застосовують механізми короткострокового або довгострокового прогнозу, зазвичай перевершують ті, що не використовують такі механізми. У свою чергу, прогностичні стратегії можна розділити на стратегії з урахуванням і без урахування причин розбалансування обчислювальної системи.

За врахуванням продуктивності вузлів стратегії розподілу поділяють на стратегії, що враховують продуктивність обчислювальних вузлів розподіленої системи та стратегії, що враховують продуктивність комунікаційної підсистеми. При врахуванні продуктивності комунікаційних підсистем обов'язковою умовою є аналіз інформаційного та керуючого трафіку обчислювальної мережі.

За використанням знань про вільні обчислювальні ресурси системи стратегії розподілу можна поділити на стратегії, які не використовують інформацію системи про наявність вільних ресурсів та стратегії, які орієнтовані на отримання таких даних у ході розподілу ресурсів. Для цих цілей, як правило, використовуються наближені та реалістичні методи оцінки кількості вільних ресурсів у системі (наприклад, кількість процесорів, обсяг доступної оперативної пам'яті, наявність вільного місця на накопичувачах для виконання різних операцій).

Як приклади стратегій, які не використовують знання про ресурси можна навести наступні стратегії розподілу: First in First Out (FIFO), Last in First Out (LIFO), Random Selection for Service, Time Sharing, Least Attained Service.

При використанні стратегії FIFO виконання завдань здійснюється у порядку їх появи у черзі. На відміну від стратегії FIFO у стратегії LIFO

виконання завдань здійснюється у зворотному порядку їх надходження у чергу. При виборі стратегії Random Selection for Service завдання будуть виконуватимуться у випадковому порядку, тобто незалежно від часу надходження їх у систему. При використанні стратегії Time Sharing слід пам'ятати, що кожному завданню, яке надходить до системи, виділяється рівномірна кількість ресурсів. Стратегія Least Attained Service передбачає, що для запуску на виконання вибирається те завдання, яке одержало найменший час обслуговування.

Слід також пам'ятати, що використання перерахованих підходів можливе лише у випадку, коли всі завдання, що надходять до системи, мають приблизно однакову обчислювальну трудомісткість. Також слід розуміти, що при розподілі слід використовувати однорідні обчислювальні вузли, і одне завдання може зайняти лише один обчислювальний ресурс.

У разі, коли є додаткова інформація (наприклад, кількісні та якісні характеристики завдань та обчислювальних ресурсів) для розподілу ресурсів можна скористатися стратегіями: Shortest Job First або Shortest Remaining processing time.

При використанні стратегії Shortest Job First для запуску на виконання обирається завдання, яке має менше вимог до ресурсів, а при використанні стратегії Shortest Remaining processing time на виконання запускається завдання, яке має мінімальний час обслуговування.

Представлена класифікація дозволяє зрозуміти принципи роботи планувальників, і навіть дає підстави вважати, що на даний час немає універсальної стратегії розподілу під усі випадки життя. Як правило, в одному програмному продукті можна використовувати кілька стратегій розподілу ресурсів.

Далі пропонується перейти до розгляду придатних для використання в гетерогенних обчислювальних середовищах приватних рішень для розподілу та балансування завдань.

1.6 Рішення для розподілу та балансування завдань

На даний час існує низка методів та алгоритмів, які орієнтовано на розподіл завдань [4-6,8,11-14]. Багато дослідників намагаються передбачити оптимальне рішення для планування запуску завдань на ресурсах кластеру. Але розподіл завдань – це, як правило, NP-повна задача, що додає додаткові труднощі при розподілі, тому що існує багато критеріїв, які впливають на розподіл в цілому. Як правило для вирішення NP-повної задачі використовують евристичні алгоритми оптимізації, крім того застосовують еволюційні алгоритми для передбачення оптимального вирішення задачі за відповідний час.

Величезна ніша у питанні планування розподілу відводиться програмам, які саме і здійснюють розподіл завдань між існуючим пулом обчислювальних ресурсів. Ці програми отримали назву – планувальники завдань, і їх основна ціль – побудувати план розподілу, що відповідає вимогам, які заявлені постачальниками завдань та обчислювальних ресурсів.

При побудові плану розподілу здійснюється оптимізація значень параметрів цільової функції [15], за рахунок яких відбувається скорочення часу виконання завдань, що в свою чергу призводить до ефективного використання обчислювальних ресурсів кластеру. Відомі планувальники мають ряд недоліків, головним з яких є орієнтація на конкретний клас задач. Сучасні планувальники, як правило, при розподілі використовують один найпростіший метод розподілу та не приймають до уваги ціну використання обчислювального ресурсу, що в свою чергу призводить до неефективного використання ресурсів кластеру.

1.6.1 Використання спрямованого ациклічного графу

У роботі [16] кожне завдання надається як ациклічний граф. Кожній його вершині відповідає окремо взята дія (задача), яка виконується під час

роботи над конкретним завданням, дуги графа визначають послідовність робіт. Граф має одну вершину-витік, що відповідає за формування вхідних даних та одну вершина-сток, яка формує результати. Вага кожної вершини графа відповідає вартості певної роботи, вага ребер графа відповідає ціні передачі з однієї вершини в іншу. Вартість виконання робіт виражається в умовних одиницях. Аналогічно формується тимчасова вартість передачі (рисунок 1.1).

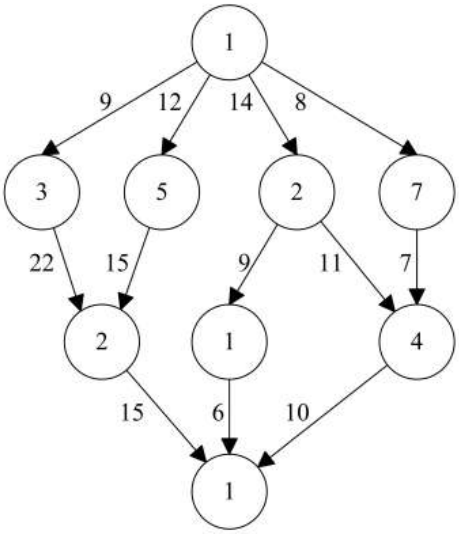

Рисунок 1.1 – Приклад ациклічного графу обчислень

Експерт визначає поняття конкретного завдання та тимчасову складність його виконання на кожному обчислювальному ресурсі (обчислювачі). Також експерт задає час передачі завдання від одного обчислювача іншому. Знаючи вартість кожної роботи, просто виконати оцінку виконання всього завдання на конкретному обчислювачі. Алгоритм упорядкування вершин DAG описано у роботі [17]. Для планування робіт використовується метод збалансованого мінімального часу завершення (Balanced Minimum Completion Time). У роботі [18] представлено математичну модель планувальника завдань, яка при розподілі враховує передісторії виконання, а також метрики завантаженості ресурсів на конкретний момент часу. Також у роботі запропоновано алгоритм пошуку найближчих сусідів, який використовує локалізоване хешування.

Характерною особливістю цього методу є облік типів атрибутів, і навіть їх значимість для ресурсопотребления.

Для порівняння з цим методом використовувався підхід, який передбачав розміщення завдань на вузлах-обчислювачах з використанням стратегії FIFO, а також підхід з пошуком найменш зайнятого обчислювального ресурсу для нових завдань на основі використання стратегії LLF (Least Loaded First). Експериментальні дані дозволили зробити такі висновки:

- загальний час обробки завдань черги скорочується на 20% при інтенсивному надходженні завдань до системи, але збільшується при середній та слабкій інтенсивності надходження на 4,5% та 10,7% відповідно;

- усереднений час очікування у черзі скорочується на 7,6% при високій інтенсивності надходження завдань, але зростає на 10,6% та 70,6% у разі середньої та слабкої інтенсивностей;

- збільшення інтенсивності надходження завдань на вхід системи дозволяє розширювати використання передісторії, що в свою чергу призводить до підвищення якості прогнозування.

1.6.2 Використання системи DIET

Система DIET [19] призначена для організації обчислень на основі різнорідних типів розподілених ресурсних об’єктів. Архітектура системи DIET (рисунок 1.2) складається із наступних компонентів:

- Client – клієнтський додаток для виконання завдань;

- Master Agent – агент, що керує. Працює із запитами від клієнта, отримує інформацію о вільних обчислювальних ресурсах, обирає необхідний та відправляє його адресу клієнту;

- Local Agent – місцевий агент. Забезпечує пердавання запросів та інформації між керуючим агентом та ресурсом, здійснює локальне планування для ресурсів;

- Server Deamon – серверна служба, яка функціонує на всіх обчислювальних ресурсах системи. Ця служба необхідна для отримання інформації о різнорідних типах задач, що вирішуються, інформації о кількості ресурсів та їх завантаженості.

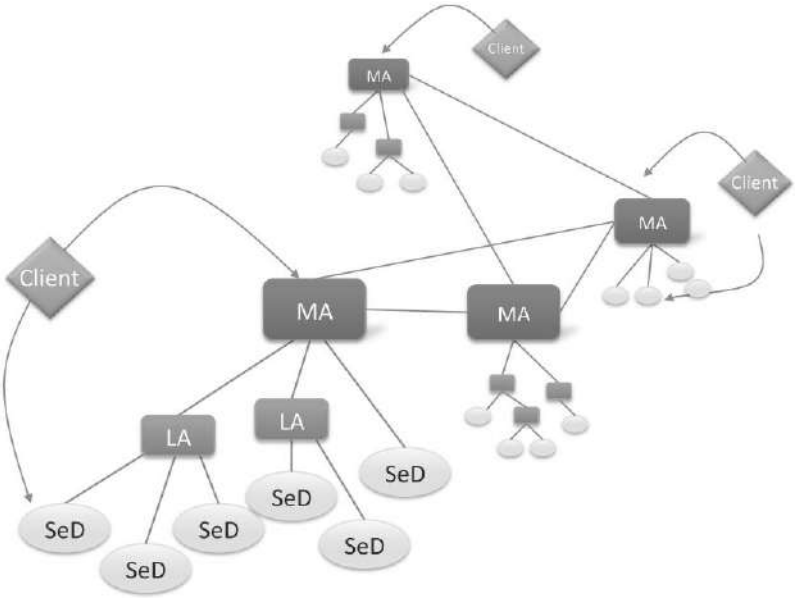

Рисунок 1.2 – Архітектура системи DIET

При надходженні на вхід системи нового завдання, серверна служба здійснює оцінювання властивостей обчислювальних ресурсів системи. Всі оцінки передаються на батьківський агент, де всі ресурси упорядковуються відповідно оптимізаційного критерію. Якщо ця функція відключена, то DIET обере обчислювальний ресурс випадковим чином.

Система DIET реалізує просту схему автоматизованого обирання обчислювальних ресурсів на базі моніторингу стану окремих вузлів системи і не використовує інформацію щодо ресурсних обмежень. Розробники DIET надають можливість модифікувати планувальник за допомогою додаткових модулів, які можливо підключити. Це дозволяє розширити множину метрик оцінювання продуктивності та задати критерії, які орієновані на упорядкування ресурсів для подальшого запуску на них завдання.

В якості недоліку системи можна виділити вимоги до програмної реалізації підпрограми планування нових прикладних додатків.

1.6.3 Використання планувальника Moab

Планувальник Moab [20] працює в режимі опитування системи контролю за вільними обчислювальними ресурсами, а також в режимі реєстрації подій. Різноманітні події можуть активувати черговий цикл планування. В якості подій можуть бути поява нового завдання на вході системи, закінчення роботи над конкретним завданням, поява в системі нового вузла (обчислювального ресурсу); коригування налаштувань (політик). Кожна ітерація планування – це звертання до системи контролю з запитом про стан обчислювальних ресурсів, рівні їх завантаження та поточних налаштувань планувальника. Далі «Moab» отримує список завдань із умовами для їх запуску та впорядковує ці завдання відповідно до їхнього відносного пріоритету. Цей пріоритет обчислюється на підставі знань про власника завдань, їх розмір, тривалість очікування в черзі та ін.

Планувальник «Moab» ґрунтується на підході з призначенням ваги для деякої кількості незалежних цілей (максимізація завантаження обчислювальних ресурсів, підвищені пріоритети окремих користувачів, виключення тривалого очікування виконання), за допомогою цих вагів обчислюється пріоритет кожного із завдань.

Основними цілями «Moab» при виборі обчислювального вузла – це підвищення продуктивності використання обчислювальних ресурсів при виконанні поточного завдання та забезпечення гнучкості планування нових завдань. Гнучкість модульного підходу до планування гарантується можливістю налаштування політики вибору вузлів виконавців як локально так і на системному рівні.

Недоліком системи слід вважати відсутність автоматизації визначення ресурсних вимог завдань, що розв'язуються. Також слід зазначити, що послідовний процес планування виконується лише для пріоритетних завдань, це призводить до локального оптимуму, тобто не найоптимальнішого рішення щодо планування виконання всіх завдань.

1.6.4 Використання планувальника Maui

Maui – це планувальник з відкритим кодом, який, як правило, використовується в паралельних кластерних системах. Планувальник Maui [21] орієнтований на роботу з UNIX подібними ОС. Як правило його використовують в паралельних та розподілених обчислювальних системах.

Перевагами планувальника Maui є використання:

- низки методів планування;
- методу Backfill;
- розширеної статистики.

Інформація про завдання, що надходять на вхід системи, містить низку атрибутів: постачальник завдання, статус завдання, наявність попередньо отриманих даних, вимоги до ресурсів, часовий інтервал, на якому потрібно їх зарезервувати, умови вибору ресурсу для запуску.

Вузол-обробник для Maui – це, як правило, набір ресурсних сутностей з безліччю пов'язаних атрибутів (процесорів, пам'яті, дискових накопичувачів, мережних карт, інтерфейсів, ліцензій на програмне забезпечення та ін.).

Інформація про ресурси (їх стан та кількість доступних) надходить до планувальника від системи управління обчислювальними ресурсами системи. Maui підтримує деякі стандартні політики планування, при їх застосуванні використовується поняття «класу» завдання, що асоціюється з одним або більшим числом атрибутів (обмежень).

Наприклад, це можуть бути атрибути завдання, які описують його тривалість чи вимоги до ресурсів; атрибути-обмеження на мінімальну/максимальну кількість вузлов-обробників; атрибути, що задають час надходження завдання, мінімальний обсяг дискового простору тощо.

Окремі класи передбачають незалежну чергу обробки. У той же час, класи відстежуються Maui як окремий вид ресурсів, що споживаються. Планувальник Maui на кожній ітерації планування виконує ті ж дії, що й Moab, проте Maui підтримує резервування ресурсів.

Недоліки системи полягають у цьому, що вона вимагає завдання точних умов вибору вузлов-обробників; розбиття завдань, що розв'язуються, на класи; завдання обмежень доступу; визначення політики роботи з чергою.

1.6.5 Використання пакетної обробки завдань

У роботі [22] представлена система, яка орієнтована на використання пластичних завдань (moldable jobs). Пластичні завдання – це завдання, які можуть бути розпаралелено, тобто вони можуть використовувати різну кількість процесорних пристроїв. Ця кількість визначає розмірністю конкретного завдання. Спочатку експерт формує список можливих розмірностей, а далі вибір необхідного значення розмірності здійснюється планувальником відповідно до умов завдання та рівнем завантаження обчислювальних ресурсів. Обов’язковою умовою є те, що всі ресурси поділяються на класи, які в свою чергу не перетинаються. Поняття «клас» поєднує ресурси з однаковою архітектурою та подібними процесорними пристроями близької продуктивності. Вузли одного класу можуть мати різну кількість оперативної та постійної пам'яті, можуть відрізнятися кількістю процесорних пристроїв та набором спеціалізованого апаратно-програмного забезпечення. Адміністратор системи бере на себе функцію опису всіх наявних ресурсів, де в якості ресурсу виступає властивість, що вирізняє об'єкти різних рівнів. Перед тим, як завдання буде відправлено на виконання, формується перелік ресурсів системи, які задовольняють вимогам, що сформовані постачальником даного завдання.

Черга завдань виникає лише у тому разі, якщо сума розмірностей наявних у системі завдань перевищує наявну кількість процесорних пристроїв. В інших випадках конкуренція завдань відсутня. Для завдань постачальник визначає список класів обчислювальних ресурсів, на яких вони можуть бути запущені, а також визначає діапазон розмірності та список необхідних ресурсів. Планувальник реалізується на централізованому клієнт-

серверному підході з сервером, модулями-агентами та клієнтськими програмами. Модулі-агенти, як і єдиний сервер представлені у вигляді фонових процесів. Агенти збирають різну інформацію про вузли та відсилають її на сервер планувальника. У подальшому вони займаються запуском завдання. Підтримується система пріоритетів завдань, що враховує загальний час простою завдань у черзі, їх складність, дані про власників та їх активність у минулому.

1.6.6 Використання потокової обробки завдань

У роботі [23] гетерогенний комплекс потокової обробки інформації представлений як розімкнена мережа масового обслуговування, що складається з перенумерованих одноканальних вузлів.

Кожен вузол-обчислювач – це апарат із відомою інтенсивністю обслуговування. Час обслуговування – це випадкова величина, яка розподілена за експонентним законом. Норма обслуговування визначається наступним чином: якщо вимога, що була отримана одним з апаратів, застає його за обчисленнями, воно ставиться в чергу очікування цього пристрою. Закінчення обробки призводить до видалення вимог з мережі.

Авторами роботи проведено дослідження моделі визначення стаціонарного розподілу ймовірностей, визначення середнього перебування вимог у системі, мінімізацію середнього перебування вимоги у системі за допомогою управління розподілом вхідного потоку. Аналіз результатів показує, що залежність ймовірності обслуговування завдань від інтенсивності потоку вимог така, що при підвищенні інтенсивності мінімальні ймовірності передачі завдань вузлу-обробнику збільшуються, а максимальні падають.

Для розглянутої моделі були досліджені метод з асинхронними обробниками, метод мінімізації часу перебування завдань у черзі та метод з підтримкою фіксованого розподілу завантаження. Експериментально

підтверджується, що метод з асинхронними обробниками найкраще підходить для вхідного потоку з високою інтенсивністю надходження заявок.

Метод мінімізації часу перебування завдань у черзі слід використовувати для малої інтенсивності вхідного потоку завдань, його не рекомендується застосовувати для випадків з великим навантаженням, так як на найбільш швидкому обробнику маємо розростання черги, деякі завдання (слабкі) простоюють.

Метод з підтримкою фіксованого розподілу завантаження найбільш ефективний як для вкрай низької, так і для вкрай високої інтенсивності вхідного потоку заявок, однак, при цьому він перевершує перші методи у широкій області середніх значень.

1.6.7 Використання мультиагентного підходу

Робота [24] присвячена огляду мультиагентного підходу із сервісно-орієнтованим управлінням розподілу завдань. У мультиагентній системі визначається низка правил групової поведінки, на основі якої виконується координація дій агентів. Агенти мають задані ролі, кожна з якої має набір правил поведінки у співтоваристві агентів. Спільнота включає агентів, які забезпечують постановку завдання, планування обчислень, класифікацію, конкретизацію та виконання завдань, моніторинг та розподіл ресурсів. Агенти можуть поєднуватися один з одним і навіть змагатися.

Аналогічний підхід був раніше представлений і роботах інших дослідників, наприклад, у матеріалах. Також був проілюстрован алгоритм управління потоками, в основі якого використовується економічна техніка управління попитом та пропозиціями щодо обчислювальних ресурсів. Ц, в свою чергу, дозволяє використовувати інформацію про політику адміністрування вузлів-обчислювачів розподіленої мережі та забезпечує необхідний рівень справедливості при управлінні ресурсами. Сформоване агентами завдання пересилається агенту-менеджеру, який взаємодіє з

агентом моніторингу ресурсів.

Така взаємодія призводить до розподілу завдань на локальні агенти обчислювачів. Завдання розподіляються за допомогою тендерної моделі, де лоти – це завдання, а учасники – це обчислювальні ресурси, що претендують на них. Описане управління може призводити до збільшення часу виконання окремих завдань, так як не завжди можливо врахувати особливості структури підзадач і користувацькі переваги щодо ресурсів.

1.7 Висновки по розділу

Методи, які реалізовано у планувальниках, використовують інформацію про ресурси системи для подальшого запуску завдань, що перекладає проблему налаштування розподілу ресурсів на кінцевого користувача (постачальника ресурсів). Облік характеру завантаження ресурсів завданнями та використання прогнозування про звільнення обчислювальних ресурсів, дозволяють слідкувати за завантаженням системи в цілому, тобто врівноважувати навантаження на кожний обчислювальний ресурс окремо.

Деякі методи показують хорошу ефективність лише при високій частоті надходження завдань на вхід системи. Експериментальні дослідження показують, що при поєднанні окремих методів із відомими стратегіями, наприклад, із FIFO або LLF дозволяє скоротити тимчасові витрати. Отже слід орієнтувати планувальних на роботу не з одним конкретним методом розподілу, а використовувати різноманітні, відповідно до типу завдань, які надходять на вхід системи.

2 РОЗРОБКА МОДЕЛІ РОЗПОДІЛУ ПУЛУ ЗАВДАНЬ НА ОБЧИСЛЮВАЛЬНІ РЕСУРСИ

2.1 Алгоритм розподілу завдань системи

Більшість завдань, які надходять в розподілену систему, можливо віднести до класу NP-повних завдань. Як правило під завданням слід розуміти програму, яку слід завантажити на обчислювальний ресурс, запустити її на виконання і після її завершення згрупувати звіт для користувача.

Всі завдання, що надійшли на вхід системи сформовані за відповідними правилами [25]: всі мають перелік вимог стосовно обчислювальних ресурсів, а також можуть мати додаткові файли опису вхідних даних для подальшого виконання. Після того як завдання надійшло до системи, воно потрапляє в чергу, з якої у подальшому може бути обрано та надіслано на відповідний обчислювальний ресурс системи.

В якості ресурсів розподіленої системи можливо використовувати різноманітні обчислювальні потужності: GRID-системи, хмари, кластера і навіть окремі ПК.

За розподіл завдань по обчислювальними ресурсами системи відповідає спеціалізований планувальний, який знаходить необхідні завданню ресурси та відслідковує наявність завдань у черзі та ресурсів у системі. На даний час існує низка планувальників (TORQUE, Portable Batch System, Condor), які працюють за встановленим правилами і використовуються в спеціалізованих середовищах моделювання. Правила розподілу завдань по обчислювальним ресурсам в даних системах, як правило, простіші: реалізація методу FIFO, LIFO, HPF, Backfill. Використання методів розподілу – це звісно перевага, але головним недоліком розподілу є те, що завдання, які надходять до системи, як правило орієновані на конкретний клас і таких класів в черзі розподіленої системи

може бути дуже багато. Ця множина класів вносить складності в процес розподілу, тому що немає універсального способу, за допомогою якого можливо розподілити всі завдання таким чином, щоб було враховано побажання і власників завдань, і власників обчислювальних ресурсів.

Виходячи з цього виникає потреба у розробці інших підходів при розподілі завдань, що надходять на вхід системи по обчислювальним ресурсам. Алгоритм розподілу пулу завдань на обчислювальні ресурси системи можливо представити наступними кроками.

Крок 1: на вхід системи надходять завдання Z_i (з відповідними вимогами до ресурсів), вони формують чергу і чекають звернення до них планувальника.

Крок 2: планувальник звертається до черги, обирає завдання Z_i і використовуючи відповідний метод розподілу підбирає обчислювальні ресурси R_j з врахуванням вимог, які надійшли від постачальників завдань. В даному випадку може виникнути три ситуації.

Ситуація №1: якщо в системі присутній обчислювальний ресурс R_j, який задовольняє вимогам завдання Z_i і він вільний, то завдання займає ресурс і виконується на ньому протягом відповідного часу. Паралельно завдання, яке знайшло обчислювальний ресурс, видаляється з черги завдань.

Ситуація №2: якщо в системі немає обчислювальних ресурсів, які потребує завдання Z_i, то буде прийнято рішення о відмові йому в обслуговуванні і завдання також покине чергу.

Ситуація №3: якщо обчислювальний ресурс присутній в системі, але він зайнятий на даний час, приймається рішення про очікування вивільнення ресурсу для подальшого запуску завдання на ньому. В даному випадку завдання залишається у черзі з поміткою про те, що воно очікує вивільнення відповідного ресурсу.

Після завершення виконання завдання, зберігаються результати, які передаються постачальнику завдань. Далі в систему надходить повідомлення про те, що ресурс біло вивільнено і він може знов приймати участь у

розподілі інших завдань системи.

Існує вирагідність, що завдання не може бути виконано за допомогою одного обчислювального ресурсу. Тоді система може прийняти одне з двох рішень.

Рішення №1: постачальнику даного завдання буде направлено повідомлення про те, що завдання не можливо виконати тими засобами, які має система.

Рішення №2: система здійснить розбивання завдання на деякі частини і кожна з них може бути направлена на виконання в систему. Система підбере множину обчислювальних ресурсів, які зможуть запустити розподілено частини завдань, а в подальшому результати кожної частини будуть зведено в єдину і завдання буде вирішено. Але такий підхід потребує додаткову інформацію від постачальника завдань (розподіл завдання на конкретні частини). Дана інформація допоможе планувальнику підібрати множину обчислювальних ресурсів таким чином, щоб скоротити час на передачу проміжних результатів між частинами завдання. Такий підхід дозволяє зменшити час перебування таких типів завдань в розподіленій системі та, в свою чергу, збільшує відсоток використання обчислювальних ресурсів та зменшує їх коефіцієнт простою.

2.2 Аналіз існуючих моделей розподілу

На даний час існує декілька моделей розподілу завдань на обчислювальні ресурси [25-27], які реалізовано в різноманітних системах розподілу та середовищах моделювання процесу розподілу. Як правило всі вони використовують лише інформацію про вимоги постачальників завдань та характеристики обчислювальних ресурсів, тобто модель розподілу має вигляд $G = \{Z, R\}$, де Z – це множина завдань, які присутні у системі, а R – відповідно множина всіх обчислювальних ресурсів. Планувальник таких систем реалізовано на основі простіших методів розподілу, які не орієнтовані

на прогнозування і ніяким чином не відслідковують ефективність використання ресурсів і час находження завдання в системі.

Існує також модель, яка використовує відповідний метод розподілу із множини методів в залежності від класу завдань [25,28], але і вона не націлена на мінімізацію часу знаходження завдань у черзі і коефіцієнт використання обчислювальних ресурсів. Така модель має наступний вигляд $G = \{Z, R, Q\}$, де Q – це множина методів розподілу.

Завдання, які приходять в систему для подальшого запуску, утворюють множину завдань, де кожне окреме завдання має низку параметрів, що будуть використовуватись у подальшому для пошуку необхідних обчислювальних ресурсів:

$$Z_i = \{ar_i^z, os_i^z, pc_i^z, ps_i^z, ms_i^z, dc_i^z, pr_i^z, ca_i^z, rt_i^z\}, \quad (2.1)$$

де ar_i – архітектура процесора (Intel, AMD);

os_i – операційна система (Windows, Linux);

pc_i – кількість процесорів;

ps_i – швидкодія процесорів;

ms_i – об'єм оперативної пам'яті;

dc_i – доступний обсяг вінчестера;

pr_i – пріоритет завдання;

ca_i – коефіцієнт зв'язності задач у завданні;

rt_i – час виконання завдання.

Для розподілу завдань необхідно постійно моніторити наявність обчислювальних ресурсів в системі R_j: наявні, зайняті або недоступні на даний час. Як і завдання обчислювальні ресурси мають низку характеристик, за якими здійснюється підбір місць запуску для завдань:

$$R_j = \{ar_j^r, os_j^r, pc_j^r, ps_j^r, ms_j^r, dc_j^r, bw_j^r, d_j^r\}, \quad (2.2)$$

де bw_j – сумарна пропускна здатність каналу (від планувальника до ресурсу) з урахуванням стану мережі на поточний час;

d_j – сумарна затримка часу передачі пакета з урахуванням стану мережі на поточний час.

Запропонована модель розподілу містить набір методів розподілу, які можливо використовувати при пошуку найкращого розподілу з точці зору завантаження ресурсів розподіленої системи.

$$Q_k = \{mn_k, lp_k\}, \tag{2.3}$$

де mn_k – метод розподілу, який використовується при пошуку відповідних ресурсів для завдань;

lp_k – низка параметрів, які враховуються при розподілі для відповідного методу (вихідні параметри).

Використання ряду методів [3,29] в планувальнику дозволяє здійснити моделювання для конкретного пулу завдань і в подальшому зробити висновок, який із методів надає найкращій розподіл (тобто з найменшим часом обробки). Однак дана модель розподілу завдань має лише один параметр, який дозволяє зробити висновок про то, що план розподілу найкращий. І цей параметр – це час виконання всього пулу завдань.

Для того, щоб розподіл завдань по обчислювальним ресурсам був більш ефективнішим слід розширити низку параметрів, які у подальшому будуть враховуватися при розробці плану використання ресурсів системи. Тому наступним кроком слід додати до системи розподілу низки критеріїв, які в подальшому дозволять використовувати потужності кластеру на всі 100%, що дозволить зменшити простій обчислювальних ресурсів і як наслідок може принести постачальнику додатковий прибуток. Також слід звернути увагу на те, що завдання, які приходять на вхід кластера для подальшого запуску мають різнорідний характер. Тому слід розробити класифікацію завдань, яка буде орієнтована на зменшення часових втрат.

2.3 Класифікація завдань відповідно об’єму даних, які треба передавати

Завдання, які приходять на вхід кластеру для подальшого запуску та виконанню можуть бути різноманітними. Тобто для кожного завдання можуть бути необхідні попередні дії, такі як передавання вхідного або вихідного об’єму даних. Таке передавання, як правило займає деякий час і його треба враховувати, тому що це може затримати початок запуску наступних завдань. Тобто треба розрізняти час, який витрачається на передавання вхідного об’єму даних, час на виконання самого завдання та час на передавання вихідного результату. І ці часи треба враховувати. В даній роботі запропоновано ввести класифікацію завдань, яка буде враховувати коефіцієнти: K_{in} – вхідний коефіцієнт та K_{out} – вихідний коефіцієнт.

Для того щоб задати такі коефіцієнти слід ввести деякі пояснення. Завдання може бути запущено на виконання тільки тоді, коли у нього є все необхідне для запуску. Тобто перед тим, як завдання запуститься пройде якийсь час, необхідний для завантаження вхідних даних: цей час позначимо як t_{in}. Далі завдання протягом якогось часу буде виконуватися і після того, як будуть отримано результати рішення, воно буде закінчено: цей час позначимо t_{work}. Також для вивантаження результатів постачальнику завдань слід мати проміжок часу: цей час позначимо t_{out}. І лише після цього завдання покине цей ресурс і він знов буду доступний для інших завдань. На основі даних значень відповідних проміжків часу сформуємо вхідний коефіцієнт (2.4):

$$K_{in} = \begin{cases} 1, & \text{якщо } t_{in} \geq t_{work} \\ 0, & \text{якщо } t_{in} < t_{work} \end{cases}. \qquad (2.4)$$

Час необхідний для завантаження початкових даний можливо розрахувати, якщо знати об’єм інформації, який треба передати та пропускну здатність каналу, через який здійснюється завантаження цієї інформації (2.5):

$$t_{in} = \frac{I_{in}}{V_{in}}, \tag{2.5}$$

де I_{in} – об'єм інформації, що завантажується;

V_{in} – пропускна здатність каналу через якій здійснюється завантаження.

Так само задамо вихідний коефіцієнт (2.6):

$$K_{out} = \begin{cases} 1, & \text{якщо } t_{out} \geq t_{work} \\ 0, & \text{якщо } t_{out} < t_{work} \end{cases}. \tag{2.6}$$

Час необхідний для вивантаження результатів також можливо розрахувати, якщо знати об'єм інформації, який треба передати та пропускну здатність каналу, через який здійснюється вивантаження цієї інформації (2.7):

$$t_{out} = \frac{I_{out}}{V_{out}}, \tag{2.7}$$

де I_{out} – об'єм інформації, що вивантажується;

V_{out} – пропускна здатність каналу через якій здійснюється вивантаження.

Завдяки введенню двох коефіцієнтів K_{in} та K_{out} можливо всі завдання, які надходять на вхід кластеру, поділити на чотири типи:

- великий об'єм вхідної інформації, великий об'єм вихідної інформації;
- малий об'єм вхідної інформації, великий об'єм вихідної інформації;
- великий об'єм вхідної інформації, малий об'єм вихідної інформації;
- малий об'єм вхідної інформації, малий об'єм вихідної інформації.

Введена класифікація дозволить планувальнику здійснювати розподіл ресурсів з урахуванням потреб завдань та зменшити часові втрати на пересилання великих об'ємів інформації. Деякі завдання можуть складатися з декількох окремих задач, які в ході виконання будуть здійснювати пересилання інформації. Якщо розмістити ці задачі не поруч, то час пересилання відповідних даних може перевищувати час виконання цих задач, що так саме негативно скажеться на ефективності обчислювальних ресурсів.

2.4 Модифікована модель розподілу пулу завдань на обчислювальні ресурси

Як говорилось раніше, аналіз розподілу за одним параметром, як правило, не дає повної картини використання ресурсів обчислювального кластеру. Тому при здійсненні розподілу слід враховувати не тільки «виграш» постачальника завдань, а й «продуктивність використання кластеру» постачальника ресурсів. Для того щоб уникнути наведеного вище недоліку рекомендується додати до моделі розподілу пулу завдань ряд показників [30], які будуть впливати на процес пошуку оптимального розподілу, а саме множину критеріїв розподілу (2.8):

$$Cr = \{t_r, t_{av}, p_d\}, \tag{2.8}$$

де t_r – загальний час виконання пулу завдань, які надійшли на вхід системи;

t_{av} – середній час очікування завдань в черзі;

p_d – відсоток простою обчислювальних ресурсів кластеру.

Наведені критерії по одинці або у зв’язці дозволять здійснювати оптимізацію відповідно за бажанням постачальників завдань та ресурсів (можливо використовувати ці критерії окремо один від одного або обирати різноманітні комбінації).

Також при розподілі слід враховувати тип завдання, який характеризується двома коефіцієнтами (K_{in} та K_{out}), які спрямовано на пошук та розподіл завдань з урахуванням об’ємів інформації, яку треба передавати на початку запуску завдання та вивантажувати результати по закінченню виконання завдання.

Після того, як до моделі розподілу вхідного пулу завдань додано низку критеріїв розподілу та коефіцієнти визначення типу завдання, вона приймає вигляд (2.10):

$$G = \{R, Z, Q, Cr, K_{in}, K_{out}\}. \tag{2.10}$$

2.5 Використання середовища моделювання GRASS

Проведення досліджень з реальними кластерними системами вимагає вкладення ресурсів (гроші, знання, час), і іноді вони можуть бути недоступні, тому дешевшим варіантом дослідження поведінки кластеру є використання імітаційного моделювання [30,31]. Імітаційне моделювання, в свою чергу, ґрунтується на побудові математичної моделі, яка в подальшому може виступати об’єктом дослідження властивостей реального кластеру. Проведений аналіз існуючих середовищ моделювання показав, що на даний час існує багато середовищ моделювання, які можливо зорієнтувати на моделювання системи, яка відповідає за розподіл завдань від користувачів на обчислювальні ресурси постачальників. Найбільшою популярністю користуються SimGrid, MicroGrid, OptorSim, SimGrid, ChicSim, GridSim [32]. Порівняльний аналіз наведених середовищ говорить про те, що більшість з них орієнтована на вузьку спеціалізацію, деякі з них мають обмеженість архітектур, а головним недоліком є те що в мережі відсутні доступні та відкрити версій цих середовищ.

За результатами проведеного аналізу було обрано середовище розподілу завдань GRASS [28]. Дане середовище дозволяє провести низку експериментів, що відтворюють процеси, які відбуваються в реальній кластерній системі. Результати, які було отримано на виході середовища, також дозволяють зробити висновки щодо ефективного використання обчислювальних ресурсів кластеру. Середовище моделювання GRASS має модульну архітектуру: ядро та модулі (плагіни), що динамічно підключаються. Кожен модуль орієнтовано на вузькоспеціалізовану задачу. При необхідності різноманітні модулі можуть бути згруповано.

Як правило планувальник будується на основі одного алгоритму, що призводить до високого відсотку простою обчислювальних ресурсів кластеру. Середовище моделювання GRASS дозволяє обирати різноманітні алгоритми розподілу, тому це і було основною перевагою його обирання. В

структурі середовища GRASS реалізовано модуль «Algoritm Loader», який оперує низкою алгоритмів розподілу. Завдяки використанню модуля «Algoritm Loader» можливо здійснити ряд експериментів, зібрати данні, а після проаналізувати результати моделювання, з подальшою рекомендацією обрати для даного пулу завдань конкретного методу їх розподілу на реальному обчислювальному кластері.

Результатом роботи середовища GRASS є множина планів розподілу завдань на обчислювальні ресурси відповідно вимогам постачальників.

2.6 Місце модуля пошуку оптимального плану розподілу завдань у середовищі GRASS

На підставі того, що у математичній моделі розподілу вхідного пулу завдань з’явилася додаткова множина (2.8) до середовища імітаційного моделювання GRASS було додано модуль, який реалізує метод пошуку оптимального плану розподілу за низкою критеріїв. Використання цих критеріїв дозволяє збільшити ефективності використання обчислювальних ресурсів кластеру за рахунок зменшення коефіцієнту простою. На рисунку 2.1 наведено місце даного модулю в середовищі GRASS.

Завдання, що надходять на вхід середовища моделювання, утворюють потік $\{Z_i\}$. Кожне завдання має дві складові: вимоги до обчислювальних ресурсів та тіло завдання (.exe файл, вхідні дані, БД тощо). На початковому етапі розподілу головне значення мають вимоги до обчислювальних ресурсів.

Паралельно із завданнями до середовища моделювання GRASS надходить інформація про доступні обчислювальні ресурси кластеру $\{R_j\}$.

Модуль вибору методів розподілу завдань містить ряд методів $\{Q_k\}$, кожен з яких використовує свій набір параметрів для розподілу. Слід звернути увагу на те, що перед запуском обчислювального експерименту можливо обрати стільки методів розподілу, скільки необхідно.

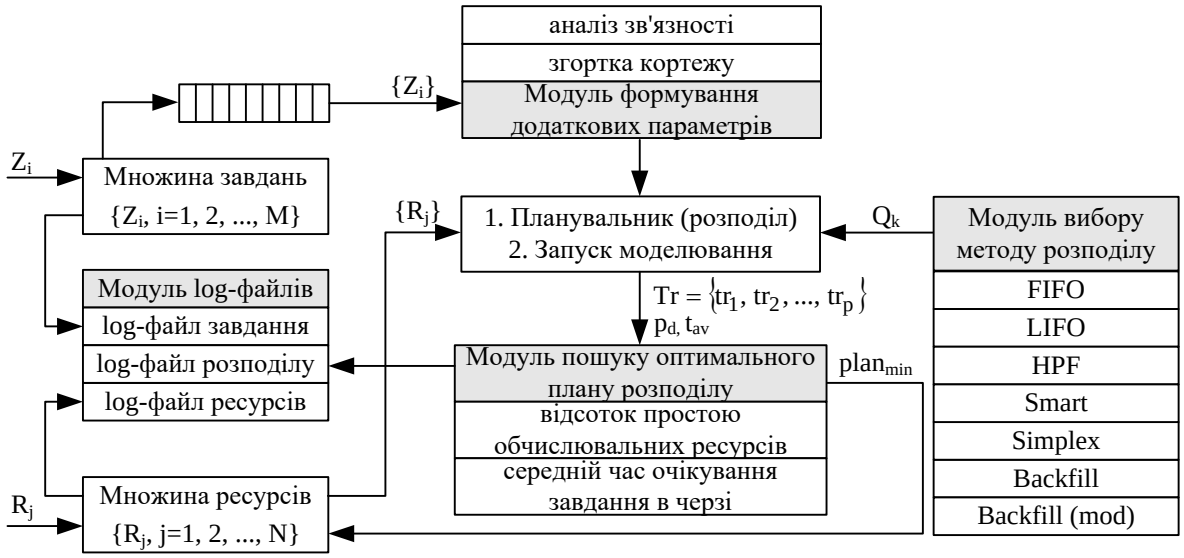

Рисунок 2.1 – Місце модулю пошуку оптимального плану розподілу в GRASS

Всі завдання, що надійшли до системи, будуть поміщено до головної черги і паралельно відбувається передача інформації по кожному завданню в модуль формування додаткових параметрів, де в подальшому буде здійснена згортка кортежу та аналіз задач у завданні на зв’язність.

При згортанні кортежу буде обчислено узагальнений критерій оцінки по кожному завданню. Таке обчислення дозволяє більш ефективно керувати процесом розподілу завдань на обчислювальні ресурси та показує, яку частину обчислювального ресурсу займає завдання в процесі його виконання [15]. Модуль формування додаткових параметрів містить два під модуля: підмодуль згортки кортежу та під модуль аналізу зв’язності задач у завданні. На виході підмодуля згортки кортежу на основі заданих правил буде сформована множина обчислювальних ресурсів, на які дане завдання може бути розподілено для подальшого запуску. Наступним шагом є аналіз зв’язності задач в завданні.

Слід зазначити що будь-яке завдання являє собою пакет задач, які об'єднано за певною тематикою. Кожна задача – це, як правило, окрема програма, що виконується. Також треба розуміти, що завдання може бути запущено на виконання тільки в тому випадку, коли для всіх його задач підібрано ресурси. Завдання можуть бути різноманітними з точки зору

зв’язності задач в ньому. Якщо задачі в завданні мають високу зв’язність (наприклад, для завдання необхідне пересилання великого обсягу даних між окремими задачами), то обирати необхідно обчислювальні ресурси, які знаходяться поруч, тобто слід зменшити час передачі даних між задачами в завданні.

Якщо в середовищі моделювання присутні обчислювальні ресурси для запуску такого завдання, але вони на даний момент зайняті, то завдання залишиться в черзі та набуде стану «Waiting». Коли необхідні ресурси будуть звільнено, завдання буде направлено для виконання на обрані обчислювальні ресурси кластеру. Якщо таких ресурсів на кластері не виявиться, то завдання набуде стану «Cancelled», далі його буде видалено з черги, з подальшим інформуванням постачальника про неможливість його виконати ресурсами даного кластеру [29]. Постачальник, отримавши таку відповідь від середовища може змінити вимоги для пошуку ресурсу для запуску завдання і знов надіслати його до середовища.

Результат роботи підмодуля аналізу зв’язності дозволяє здійснювати підбір обчислювальних ресурсів на кластері з урахуванням зменшення часу на передачу вхідних та вихідних даних для завдання.

Після того як закінчив роботу модуль формування додаткових параметрів планувальник може починати розподіл завдань із пулу на обчислювальні ресурси, які на даний час присутні у системі. Для розподілу завдань на обчислювальні ресурси кластеру планувальнику необхідні наступні дані:

- інформація про обчислювальні ресурси, які присутні в кластері на даний час;

- інформація про обчислювальні ресурси, що на даний час є задіяними та час їх вивільнення;

- метод розподілу (один або декілька);

- інформація про зв’язність задач в завданні.

На основі даних, які було отримано, планувальник починає будувати

план розподілу для кожного завдання на обчислювальні ресурси кластеру. Результатом роботи планувальника є множина планів розподілу за обраними заздалегідь методами розподілу.

Далі плани розподілу, які було сформовано планувальником, запускаються в середовище моделювання з метою отримання результатів моделювання. Після закінчення процесу моделювання результати можна буде проаналізувати та обрати оптимальний план розподілу для конкретного вхідного пулу завдань. В наслідок того, що в математичну модель було додано множина критеріїв розподілу після закінчення процесу моделювання з'являється множина часів виконання пулу завдань t_r за кожним методом розподілу, а також значення критеріїв, які також вираховуються для кожного методу (t_{av} та p_d).

Наступним кроком є запуск методу для пошуку оптимального рішення для конкретного вхідного пулу завдань. Робота цього методу полягає в аналізі отриманих планів розподілу з урахуванням критеріїв розподілу, які було задано постачальниками. Ці данні (вагові коефіцієнти) постачальники завдань та ресурсів вносять до імітаційного середовища. Вагові коефіцієнти необхідні для розрахунку критерію оцінки розподілу та подальшого аналізу плану розподілу. Робота методу пошуку оптимального рішення здійснюється в декілька кроків.

На першому кроці здійснюється завантаження вхідних даних:

- множини часів виконання планів розподілу $Tr = \{tr_1, tr_2, ..., tr_p\}$, $\forall p = 1..P$ по кожному методу розподілу, який було обрано $(mn \in Q)$ $f : Plan_k \rightarrow Tr_k$;

- відсотку простою ресурсів кластеру p_d по кожному із обраних методів розподілу;

- середнього часу очікування кожного завдання в черзі t_{av} по кожному із обраних методів розподілу.

На другому кроці здійснюється саме аналізу планів розподілу на

підставі використання параметрів часткових критеріїв: t_r, t_{av} та p_d. Плани розподілу можливо аналізувати по конкретному критерії, або використовувати критерії в комбінації, але в даному випадку їх спочатку слід об’єднати в узагальнений (адитивний) критерій [15]. Після цього, як критерії об’єднано, можливо здійснювати подальший аналіз в напрямку пошуку оптимального розподілу по конкретному пулу вхідних завдань.

На виході модулю пошуку оптимального плану розподілу формується рекомендаційна інформація щодо кожного плану розподілу, яка пересилається постачальникам завдань та ресурсів з відповідними розрахунками та рекомендаціями.

Будь-яке завдання, яке надійшло до середовища моделювання, буде обов’язково розподілено. Винятком може бути лише той випадок, коли в системі на даний час не знайдеться необхідних обчислювальних ресурсів. У цьому разі, нажаль, завданню буде відмовлено у виконанні, і воно буде відправлено постачальнику для зміни вимог.

3 АНАЛІЗ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ

3.1 Використання узагальнених критеріїв

Будь-яке завдання, що надходить на вхід кластеру для подальшого виконання описується вимогами до обчислювальних ресурсів, які йому необхідні. Таких вимог може бути багато, але не зовсім зрозуміло, за яким із них здійснювати оптимізацію. На даний час існують завдання, які необхідно аналізувати на низкою параметрів, але використання декількох параметрів при оптимізації накладає додаткові труднощі для процесу розподілу. Слід зазначити, що іноді не зовсім зрозуміло, яким чином використовувати декілька параметрів, який із них буде головним, а який уточнювальним.

Якщо розглянути характеристики завдання (2.1), то можливо умовно розділити на два класи: якісні характеристики та кількісні. Якісні характеристики характеризують завдання у цілому, і їх значення можна представити у вигляді множини дискретних значень. Тому узагальненим коефіцієнтом для якісних параметрів зручно прийняти мультиплікативний коефіцієнт.

Кількісні характеристики – це фізичні значення, які для завдань є необхідними, а для обчислювальних ресурсів – певними, тобто заданими. Тому для кількісних характеристик в якості узагальненого коефіцієнту слід обрати адитивний коефіцієнт.

Метод рішення багатокритеріальних задач оптимізації з використанням узагальненого (інтегрального) критерію заснований на об’єднанні приватних критеріїв $F_i(X)$, $i = \overline{1, n}$ в один інтегрований $F(X) = \Phi(F_1(X), F_2(X), ..., F_n(X))$ з подальшим знаходженням максимуму або мінімуму даного критерію [15].

При використанні адитивного критерію цільова функція у загальному вигляді має вигляд (3.1):

$$F(X) = \sum_{i=1}^{n} C_i \frac{F_i(X)}{F_i^0(X)} = \sum_{i=1}^{n} C_i f_i(X) \to \max(\min), \quad (3.1)$$

де n – кількість приватних критеріїв, що об'єднуються;

C_i – ваговий коефіцієнт і-го приватного критерію;

$F_i(X)$ – числове значення і-го приватного критерію;

$F_i^0(X)$ – і-й дільник, що нормує;

$f_i(X)$ – нормоване значення і-го приватного критерію.

В якості дільників, що нормують, приймаються директивні значення критеріїв, які було задано постачальниками або максимальні (мінімальні) значення критеріїв, що досягаються в області допустимих рішень. Також слід розуміти, що узагальнений адитивний критерій, який отримується в результаті розрахунків є безрозмірною величиною.

3.2 Вхідні значення та результати моделювання

Перед початком моделювання до імітаційного середовища було завантажено пули вхідних завдань та обчислювальних ресурсів кластеру. Пули завдань та ресурсів було сформовано виходячи із правил (2.1, 2.2). Для обрання оптимального методу розподілу завдань було задано наступні вагові коефіцієнти для приватних параметрів $t_r = 0.4$, $p_d = 0.3$, $t_{av} = 0.3$.

Використовуючи набір методів розподілу, які були присутні в імітаційному середовищі GRASS (FCFS, LIFO, HPF, Backfill, Backfill_mod, Simplex та Smart), було проведено імітаційне моделювання та отримані відповідні результати (таблиця 3.1 та рисунки 3.1-3.3). На рисунку 3.1 відображено значення сумарного часу виконання вхідного пулу завдань для кожного методу розподілу, який присутній у середовищі GRASS. На рисунку 3.2 відображено значення часу, який завдання провели у черзі, а на рисунку 3.3 відображено коефіцієнт простою обчислювальних ресурсів кластеру.

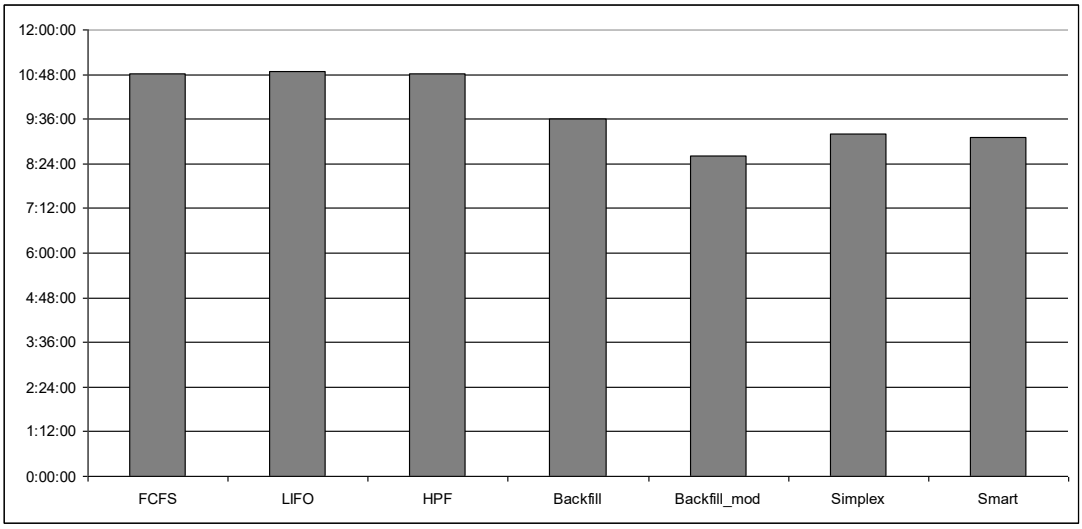

Рисунок 3.1 – Час виконання пулу завдань по кожному методу розподілу

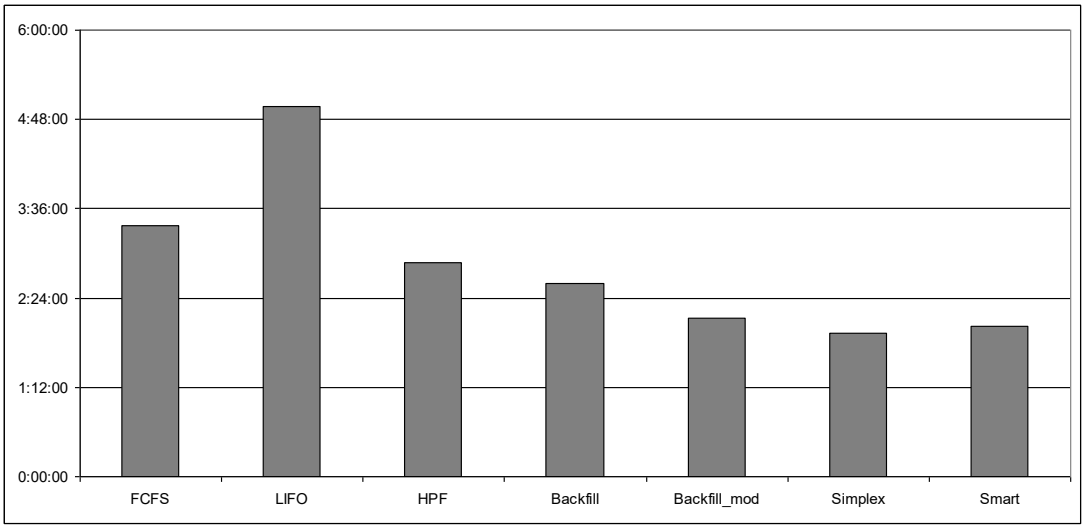

Рисунок 3.2 – Час знаходження завдань в черзі по кожному методу розподілу

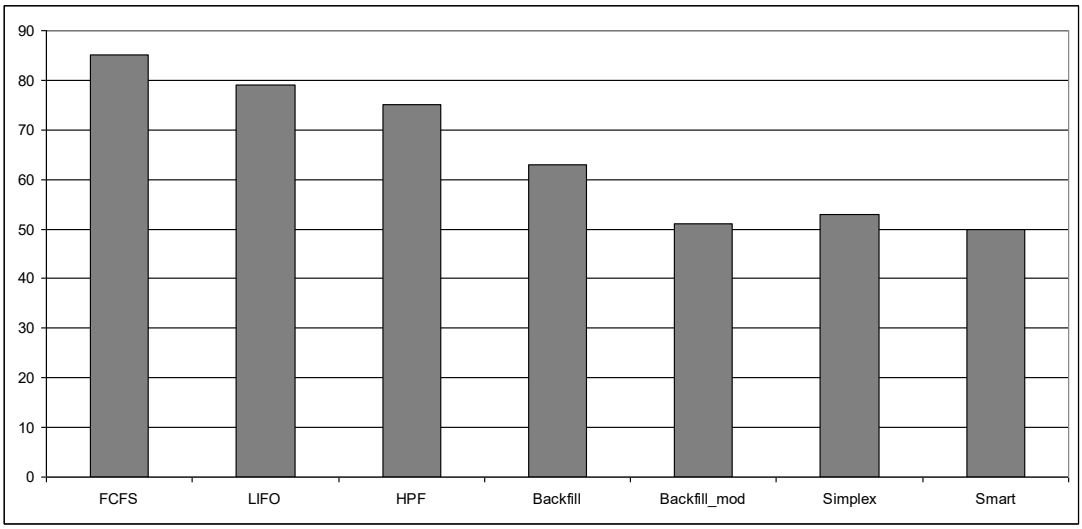

Рисунок 3.3 – Коефіцієнт простою ресурсів по кожному методу розподілу

Таблиця 3.1 – Результати моделювання за одним критерієм

Методи розподілу	Значення вагових коефіцієнтів C_i для критеріїв F_i		
	$t_r = 0.4$	$t_{av} = 0.3$	$p_d = 0.3$
FCFS	10:48:55	3:22:32	85
LIFO	10:52:48	4:58:24	79
HPF	10:48:37	2:52:48	75
Backfill	9:36:43	2:35:59	63
Backfill_mod	**8:36:56**	2:07:44	51
Simplex	9:12:11	**1:55:54**	53
Smart	9:05:54	2:01:37	**50**

3.3 Аналіз результатів моделювання

За результатами таблиці 3.1 можливо зробити висновок про те, що в залежності від того, який критерій розподілу було обрано в якості головного – результат буде різний:

- якщо в якості критерію оптимізації було обрано $t_r \to \min$, то для розподілу слід обрати алгоритм розподілу Backfill_mod, тому що завдяки йому вхідний пул завдань буде виконано за 8:36:56 (це і є найкращій результат);

- якщо в якості критерію оптимізації було обрано $t_{av} \to \min$, то для розподілу слід обрати алгоритм розподілу Simplex, тому що завдяки йому середній час очікування завдання в черзі складає мише 1:55:54 (це і є найкращій результат);

- якщо в якості критерію оптимізації було обрано $p_d \to \min$, то для розподілу слід обрати алгоритм розподілу Smart, тому що він дає найкращий результат – 50% (це і є найкращій результат).

Але все буде по-іншому, якщо використовувати комбінації критеріїв розподілу (t_r, t_{av}, p_d). Результат такого аналізу наведено у таблиці 3.2.

Таблиця 3.2 – Результат аналізу за комбінацією критеріїв

Методи розподілу	Комбінації критеріїв розподілу			
	$t_r\ p_d$	$p_d\ t_{av}$	$t_r\ t_{av}$	$t_r\ p_d\ t_{av}$
FCFS	1,01213	1,03425	1,02637	1,53637
LIFO	0,97913	1,24639	1,27752	1,75152
HPF	0,9519	0,89728	0,94918	1,39918
Backfill	0,8243	0,7818	0,85	1,228
Backfill_mod	**0,706**	0,63662	0,73062	**1,03663**
Simplex	0,7453	0,618	**0,7273**	1,0453
Smart	0,7224	**0,6148**	0,7372	1,0372

Якщо в якості критеріїв розподілу була обрана комбінація критеріїв $\{t_r, p_d\} \to \min$, то вхідні завдання слід розподіляти використовуючи алгоритм розподілу Backfill_mod, тому що даний алгоритм надає найменше значення – 0,706.

Якщо в якості критеріїв розподілу була обрана комбінація критеріїв $\{p_d, t_{av}\} \to \min$, то вхідні завдання слід розподіляти використовуючи алгоритм розподілу Smart тому що даний алгоритм надає найменше значення – 0,6148.

Якщо в якості критеріїв розподілу була обрана комбінація критеріїв $\{t_r, t_{av}\} \to \min$, то вхідні завдання слід розподіляти використовуючи алгоритм розподілу Simplex, тому що даний алгоритм надає найменше значення – 0,7273.

Якщо в якості критеріїв розподілу була обрана комбінація критеріїв $\{t_r, p_d, t_{av}\} \to \min$, то вхідні завдання слід розподіляти використовуючи алгоритм розподілу Backfill_mod, тому що даний алгоритм надає найменше значення – 1,03663.

Всі результати отримано завдяки розрахункам, які стосуються конкретних пулів завдань та обчислювальних ресурсів. Але якщо обрати інші

пули завдань та ресурсів для моделювання, а також інші числові значення вагових коефіцієнтів для приватних параметрів, то результати пошуку найкращого методу розподілу будуть інші.

Наступним кроком моделювання була перевірка залежності виконання часу пулу завдань, що надійшли на вхід кластеру від об’єму даних, які передаються на обчислювальний ресурс перед запуском завдання. В таблиці 3.3 наведено результати моделювання відповідно до кожного з чотирьох типів завдань та пулу завдань, якій містить завдання різноманітних типів.

Таблиця 3.3 – Загальний час виконання пулу завдань відповідно їх типів

Методи розподілу	Типи завдань, час				
	1-й	2-й	3-й	4-й	загальний
FCFS	20:45:38	16:55:43	17:58:49	12:25:48	17:01:30
LIFO	20:38:43	16:25:29	16:09:28	12:27:55	16:25:24
HPF	19:21:59	15:36:46	16:03:24	**8:15:33**	14:49:25
Backfill	15:47:14	11:26:15	**11:49:27**	8:51:37	11:58:38
Backfill_mod	14:22:55	**10:38:09**	12:09:01	8:26:51	**11:24:14**
Simplex	15:57:32	11:43:18	12:38:31	8:25:44	12:11:16
Smart	**14:12:48**	10:55:57	11:29:46	9:02:26	11:25:14

Результат моделювання загального часу виконання пулу завдань відповідно їх типів наведено на рисунках 3.3 та 3.4. На рисунку 3.4 завдання згруповано відповідно методів, які використовуються в середовищі GRASS. За результатами моделювання, що наведено на рисунку 3.4, можливо зробити висновок, що пул завдань, який містить завдання 4-го типу виконується швидше усіх інших. Це пов’язано з тим, що дані завдання не використовують обчислювальні ресурси протягом завантаження та вивантаження інформації. Рисунок 3.5 ілюструє ті ж самі результати моделювання, але вони згруповані відповідно типу.

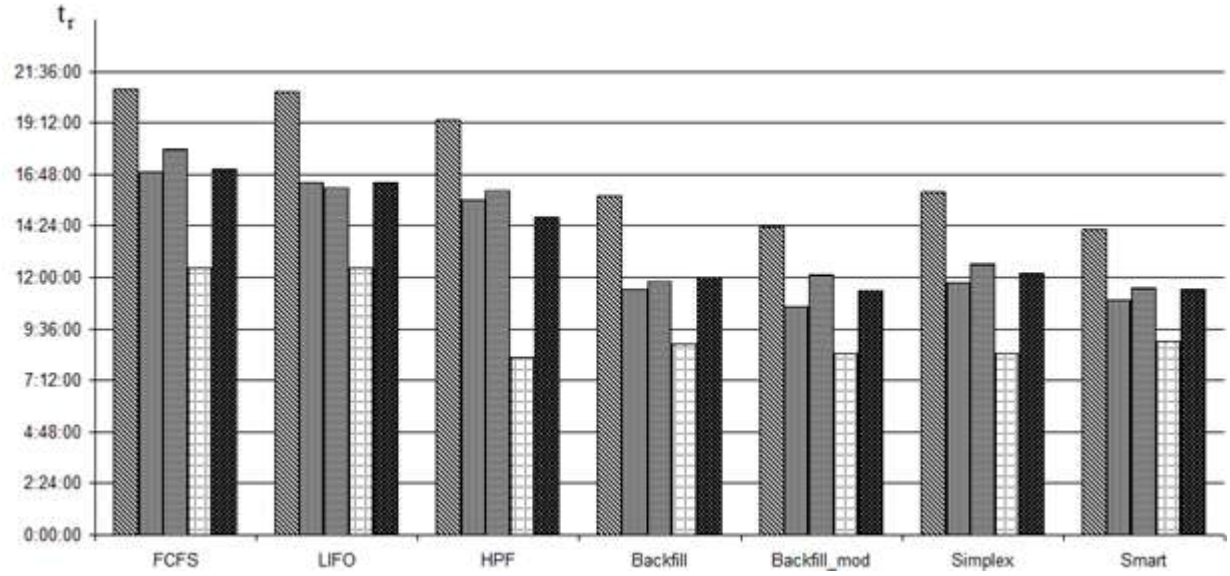

Рисунок 3.4 – Загальний час виконання пулу завдань по методам розподілу

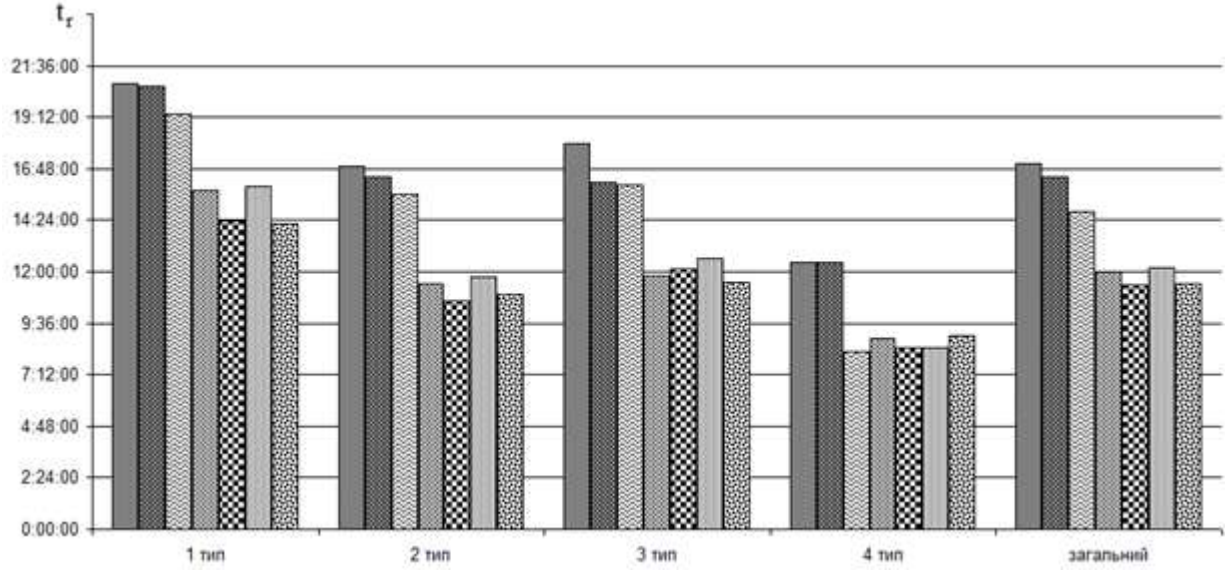

Рисунок 3.5 – Загальний час виконання пулу завдань відповідно до їх типів

Наступний крок моделювання пов’язаний з необхідністю проаналізувати середній час очікування завдань вхідного пулу у черзі (таблиця 3.4). Рисунки 3.6 та 3.7 ілюструють результати моделювання. За результатами моделювання, які наведено на рисунку 3.6 можна побачити, що неможливо обрати якийсь один тип завдань, який при використанні різних методів надасть найкращій результат. Але можливо зробити висновок, що розподіляти завдання за методами FCFS та LIFO не слід. Наведені методи в середовищі GRASS використовуються для наочності процесу розподілу.

Таблиця 3.4 – Середній час очікування завдань в черзі відповідно їх типів

Методи розподілу	Типи завдань, час				
	1-й	2-й	3-й	4-й	загальний
FCFS	6:54:23	4:26:41	3:58:35	3:45:22	4:46:15
LIFO	6:43:59	4:57:31	3:49:57	3:54:48	4:51:34
HPF	4:57:38	3:24:42	2:59:46	3:28:27	3:42:38
Backfill	3:39:41	**3:02:28**	2:24:21	2:59:29	3:01:30
Backfill_mod	**3:02:14**	3:18:09	2:18:51	2:37:14	**2:49:07**
Simplex	3:42:23	3:20:37	2:45:57	**2:19:24**	3:02:05
Smart	3:51:57	3:29:57	**1:58:16**	3:07:23	3:06:53

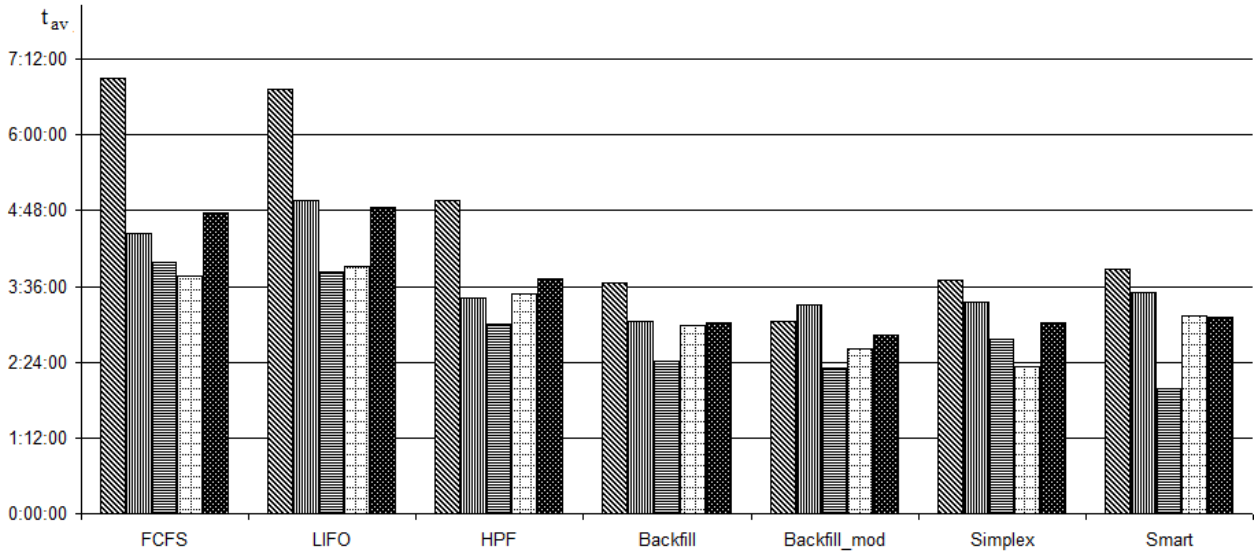

Рисунок 3.6 – Середній час очікування завдань в черзі по методам розподілу

Рисунок 3.7 ілюструє, що для першого типу завдань час надходження завдань в черзі вище для всіх інших методів та типів. Також слід зазначити, що перший тип завдань надає найкращий варіант розподілу, коли використовується метод Backfill_mod, а другий, коли використовується метод Backfill. Третій тип завдань надає кращий варіант при використанні методу Smart, а для четвертого типу завдань слід обрати Simplex. Коли завдання в пулі мають різні типи, обирати слід метод Backfill_mod [12] (але цей варіант лише для пулу, який використовували при моделюванні).

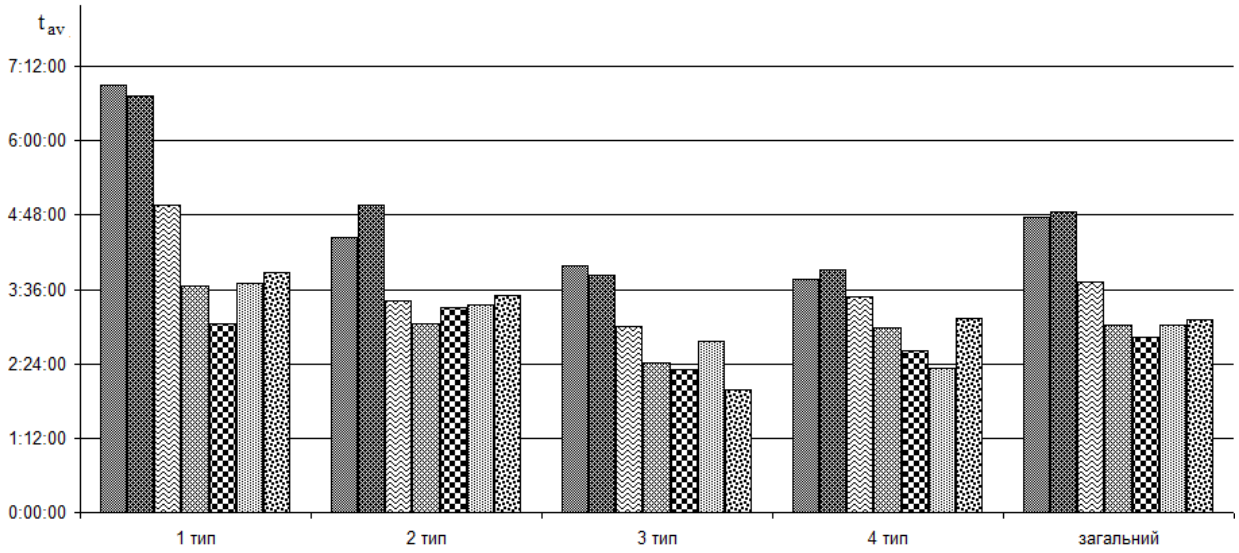

Рисунок 3.7 – Середній час очікування завдань в черзі по типам завдань

Далі необхідно здійснити моделювання, яке орієнтовано на необхідність проаналізувати відсоток простою обчислювальних ресурсів кластеру за типом завдань, які надходять на вхід кластеру (таблиця 3.5). Рисунки 3.8 та 3.9 ілюструють результати моделювання. Аналіз даних, які було отримано, не надають відповідь, який тип завдання надає кращий результат розподілу. Всі методи розподілу, які присутні у середовищі видають результат в межах від 55 до 89%. Більш стабільний результат показує пул, якій містить завдання різноманітних типів.

Таблиця 3.5 – Відсоток простою обчислювальних ресурсів кластеру за типом

Методи розподілу	Типи завдань, %				
	1-й	2-й	3-й	4-й	загальний
FCFS	85	89	88	80	86
LIFO	87	88	85	82	86
HPF	79	83	85	76	81
Backfill	70	74	**68**	66	70
Backfill_mod	67	72	70	**55**	**63**
Simplex	**64**	77	72	67	70
Smart	71	**68**	73	63	69

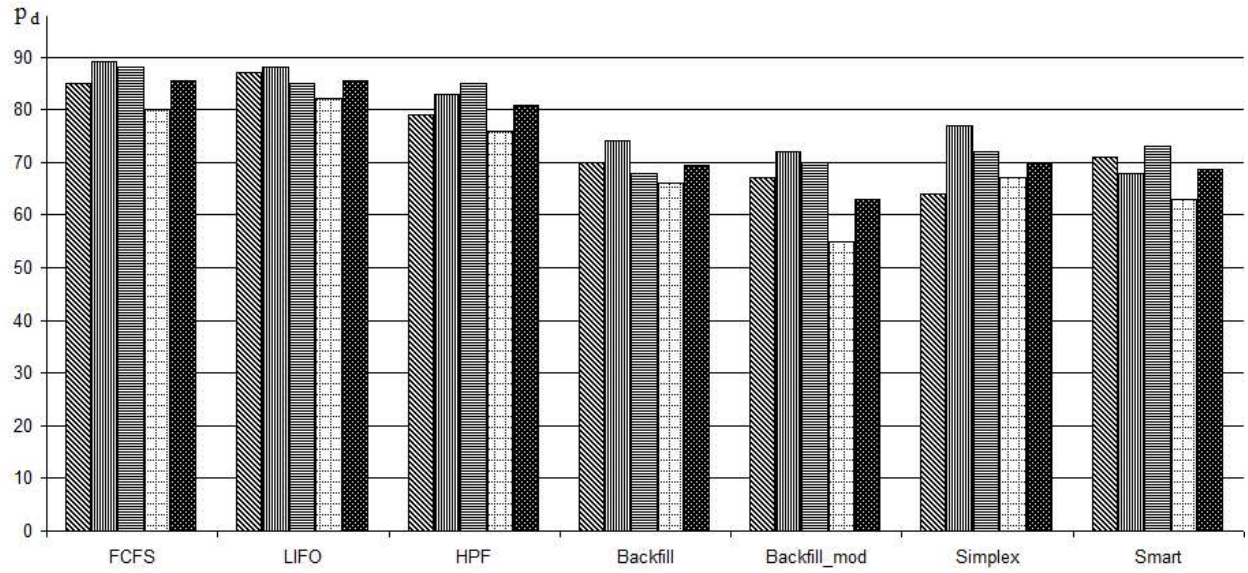

Рисунок 3.8 – Відсоток простою обчислювальних ресурсів кластеру відповідно методів розподілу

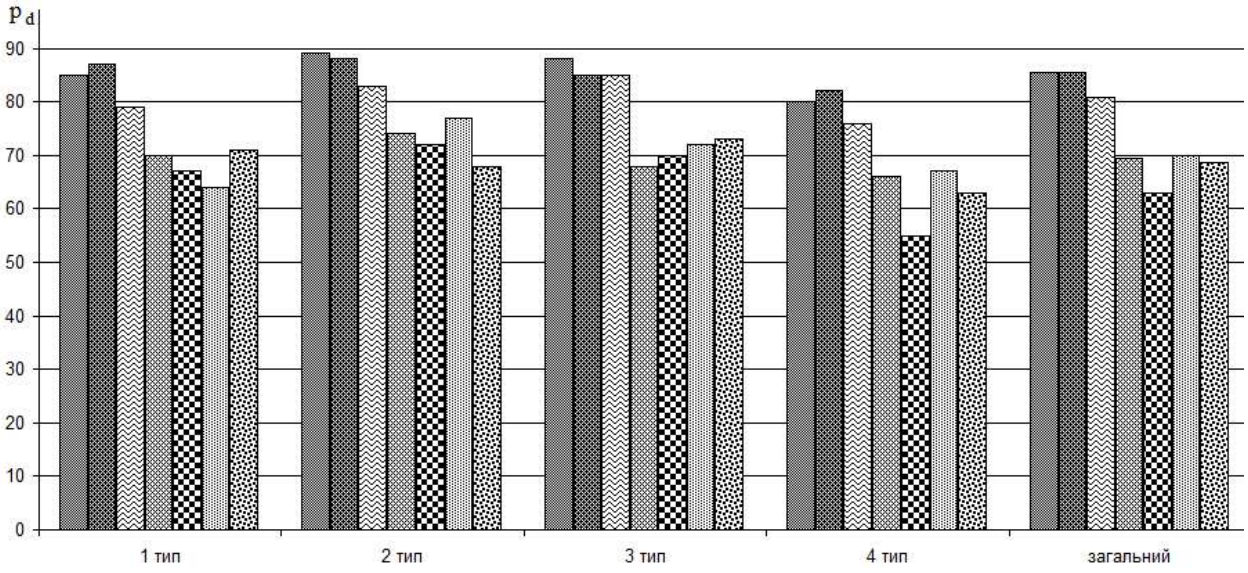

Рисунок 3.9 – Відсоток простою обчислювальних ресурсів кластеру відповідно типам завдань

Наступний експеримент пов'язаний з пошуком найкращого плану розподілу відповідно за комбінаціями критеріїв. Як було зазначено вище, розподіл за одним критерієм не надає повної картини, яка існує на даний час у кластері. Також використання комбінацій критеріїв дозволить задовольнити побажання як постачальників завдань, так і постачальників обчислювальних ресурсів. Результати розрахунку, які спрямовані на пошук

найкращого розподілу наведено у таблицях 3.7-3.10.

Розрахунок здійснюється з використанням (інтегрального) адитивного критерію (3.1). Наприклад, найкращим варіантом при використанні двох критеріїв (загального часу виконання пулу завдань та відсотку простою обчислювальних ресурсів) є той з них, хто має найменше значення цих параметрів. В якості вихідних значень виступають значення, які наведено у таблицях 3.3-3.5.

Розглянемо більш докладно процес розрахунку на прикладі комбінації критеріїв розподілу t_r p_d для завдань першого типу. Сформуємо відповідну таблицю вхідних значень (таблиця 3.6).

Таблиця 3.6 – Вхідні дані для розрахунку адитивного критерію для t_r p_d

Критерій розподілу	Вагові коефіцієнти	Значення параметрів для методів розподілу						
		FCFS	LIFO	HPF	Backfill	Backfill_mod	Simplex	Smart
t_r	0,4	20:45:38	20:38:43	19:21:59	15:47:14	14:22:55	15:57:32	**14:12:48**
t_{av}	0,3	6:54:23	6:43:59	4:57:38	3:39:41	**3:02:14**	3:42:23	3:51:57
p_d	0,3	85	87	79	70	67	**64**	71

При використанні адитивного критерію цільова функція у загальному вигляді буде виглядати як $F(X)=\sum_{i=1}^{n} C_i \frac{F_i(X)}{F_i^0(X)} = \sum_{i=1}^{n} C_i f_i(X) \to \min$. В якості дільників, що нормують, для даного приклада візьмемо найменше значення приватних критеріїв: $F_{t_r}^0(X)=14:12:48$ та $F_{t_{av}}^0(X)=64$. Значення адитивного критерію розрахуємо відповідно для кожного методу розподілу.

FCFS: $F(X)=0.4\cdot\left(\frac{20:45:38}{14:12:48}\right)+0.3\cdot\left(\frac{85}{64}\right)=1.26643;$

LIFO: $F(X) = 0.4 \cdot \left(\frac{20:38:43}{14:12:48}\right) + 0.3 \cdot \left(\frac{87}{64}\right) = 1.246066;$

HPF: $F(X) = 0.4 \cdot \left(\frac{19:21:59}{14:12:48}\right) + 0.3 \cdot \left(\frac{79}{64}\right) = 1.034997;$

Backfill: $F(X) = 0.4 \cdot \left(\frac{15:47:14}{14:12:48}\right) + 0.3 \cdot \left(\frac{70}{64}\right) = 0.80595;$

Backfill_mod: $F(X) = 0.4 \cdot \left(\frac{14:22:55}{14:12:48}\right) + 0.3 \cdot \left(\frac{67}{64}\right) = 0.70475;$

Simplex: $F(X) = 0.4 \cdot \left(\frac{15:57:32}{14:12:48}\right) + 0.3 \cdot \left(\frac{64}{64}\right) = 0.81522;$

Smart: $F(X) = 0.4 \cdot \left(\frac{14:12:48}{14:12:48}\right) + 0.3 \cdot \left(\frac{71}{64}\right) = 0.78185.$

Як бачимо із розрахунків, найкращим є п’ятий варіант (тобто метод Backfill_mod), тому що йому відповідає мінімальне значення інтегрального адитивного критерію.

Далі здійснимо розрахунок для всіх інших типів завдань по кожному методу, результати занесемо до таблиці 3.7. Рисунок 3.10 ілюструє результати розподілу пулу вхідних завдань по кожному методу розподілу. Зрозуміло, що варіант, коли завдання пулу будуть мати однаковий тип завдань, це межовий результат. Річ у тому, що передбачити, які завдання будуть формувати пул неможливо, тому що вони, як правило, носять гетерогенний (розрізнений) характер. Але користуючись даними результатами можливо зробити висновок, що методами, які можуть надати найкращий результат можуть бути Simplex (метод, який використовує пошук відповідно до задач лінійного програмування), Smart (метод, який орієнтовано на перші ресурси, які будуть вивільнено), Backfill (метод зворотного заповнення) та Backfill_mod (метод, який в доповнення до Backfill здійснює врахування пропускної здатності каналу та його ширини).

В таблиці 3.7 типи завдань по кожному методу розподілу виділено різними кольорами, рисунок 3.10 також має відповідну кольорову складову.

Таблиця 3.7 – Комбінації критеріїв розподілу t_r p_d

Методи розподілу	Типи завдань				
	1-й	2-й	3-й	4-й	загальний
FCFS	1,26643065	1,07512691	1,21345533	1,08700495	1,1049503
LIFO	1,2460655	1,1068705	1,1299023	1,1090148	1,0932777
HPF	1,0349965	0,9237311	0,9991847	0,8486011	0,9148838
Backfill	0,805945	0,7301497	0,766164	0,815375	0,7420772
Backfill_mod	**0,7047452**	**0,7257855**	0,763245	0,7475	**0,7**
Simplex	0,815221	0,770678	0,84862	**0,70822**	0,750497
Smart	0,766097	0,756344	**0,688902**	0,841107	0,732101

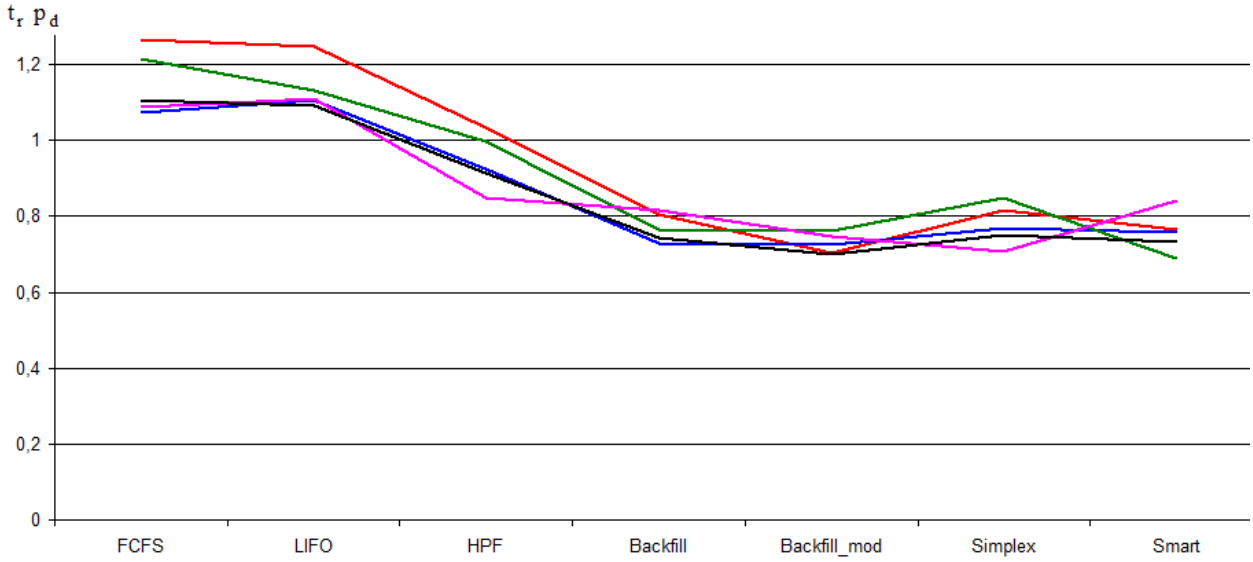

Рисунок 3.10 – Розподіл вхідного пулу завдань на обчислювальні ресурси кластеру відповідно критеріям t_r та p_d

Після того, як математичну модель було модифіковано за допомогою критеріїв розподілу t_r, t_{av} та p_d можливо розподіляти вхідні завдання на обчислювальні ресурси кластеру не лише за одним із них, а використовувати їх в зв'язці, що в свою чергу надасть більший простір для оптимізації як бажань постачальників ресурсів, так і бажань постачальників завдань. У прикладі, який було розглянуто вище, використовувалась пара критеріїв t_r та p_d. Так само можливо використовувати ці критерії і в іншій комбінації.

Наприклад, пари: t_{av} та p_d, t_r та t_{av}, або навіть всі три: t_r, t_{av} та p_d. Інші результаті розрахунків наведено в таблицях 3.8-3.10, а також проілюстровано на рисунках 3.11-3.13. Результати моделювання показують, що використання методів FCFS та LIFO ніколи нажаль не дають оптимальних рішень, і ці алгоритми розподілу було використано в якості порівняння з більш ефективними методами.

Таблиця 3.8 – Комбінації критеріїв розподілу p_d t_{av}

Методи розподілу	Типи завдань				
	1-й	2-й	3-й	4-й	загальний
FCFS	0,947799867	0,7002283	0,86402364	1,08700495	0,7808014
LIFO	0,936929	0,7479814	0,8333005	1,1090148	0,7902328
HPF	0,7368512	0,5806723	0,7060034	0,8486011	0,6520773
Backfill	0,5804017	**0,5176471**	0,566164	0,815375	0,5441893
Backfill_mod	**0,509375**	0,5375502	0,558095	0,7475	**0,5**
Simplex	0,566097	0,556312	0,63272	**0,70822**	0,545224
Smart	0,587972	0,545186	**0,514706**	0,841107	0,550564

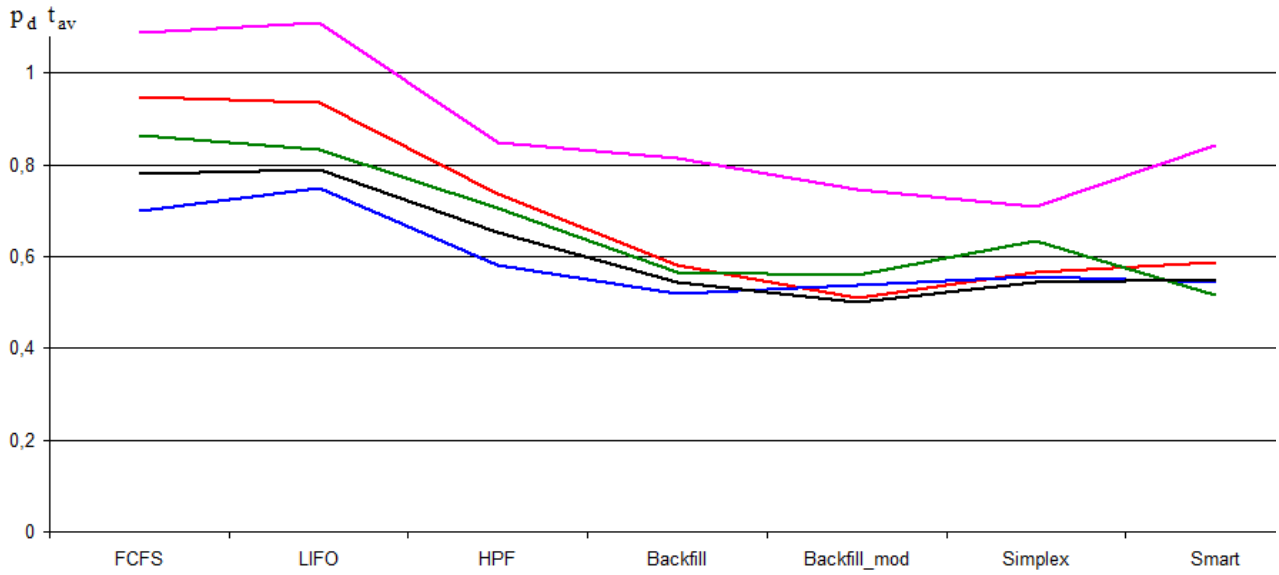

Рисунок 3.11 – Розподіл вхідного пулу завдань на обчислювальні ресурси кластеру відповідно критеріям t_{av} та p_d

Таблиця 3.9 – Комбінації критеріїв розподілу t_r t_{av}

Методи розподілу	Типи завдань				
	1-й	2-й	3-й	4-й	загальний
FCFS	0,849880785	0,8984279	0,86707875	0,89290687	0,8701806
LIFO	0,8528866	0,8765362	0,7966018	0,9018881	0,8490767
HPF	0,7918953	0,8312941	0,7931813	0,6763636	0,7770922
Backfill	0,7918953	0,6477967	**0,6**	0,6691124	0,6423323
Backfill_mod	**0,6141202**	0,6117647	0,616914	**0,609121**	**0,6**
Simplex	0,649124	0,667307	0,639429	0,651856	0,649718
Smart	0,621875	**0,611157**	0,603608	0,666934	0,619632

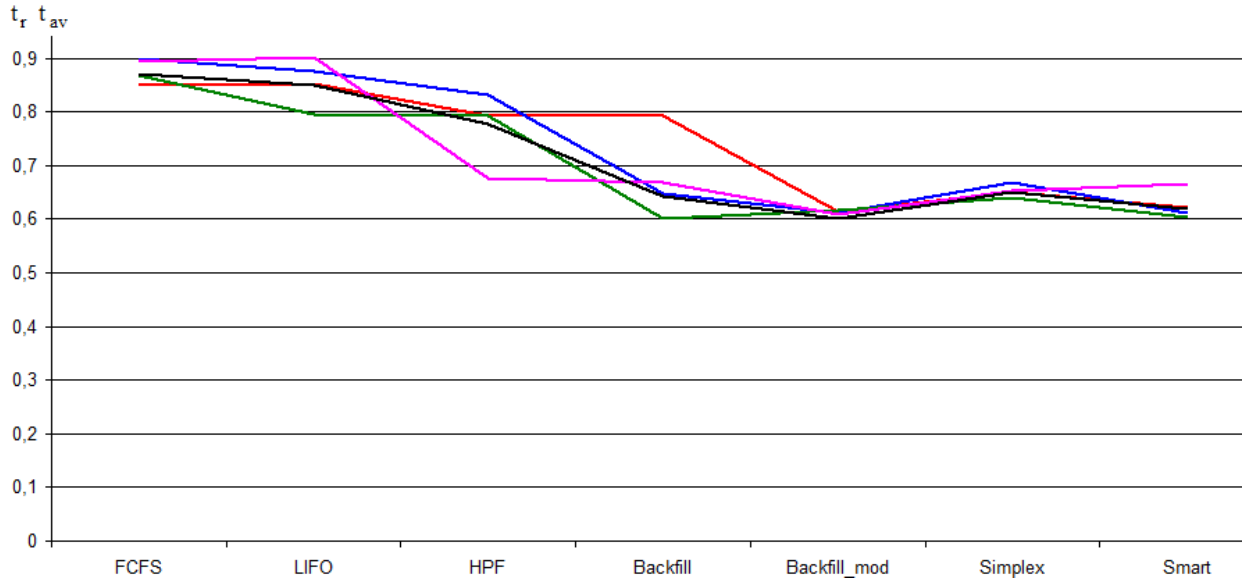

Рисунок 3.12 – Розподіл вхідного пулу завдань на обчислювальні ресурси кластеру відповідно критеріям t_r та t_r

Результати, які було отримано протягом експерименту, що використовує вісі три критерії розподілу у зв'язці, рекомендують обирати для розподілу метод Backfill_mod. Такий результат отримана завдяки тому, що використання даного методу орієнтовано не лише на вимоги завдання до обчислювальних ресурсів, а й на те що при розподілі буде враховано розташування всіх складових завдання. Це врахування, в свою чергу, дозволить знизити час, який задачі завдання будуть використовувати на

пересилання інформації між собою протягом виконання, а також пропускну здатність та ширину каналу зв'язку для пересилання вхідного та вихідного об'єму інформації.

Таблиця 3.10 – Комбінації критеріїв розподілу t_r p_d t_{av}

Методи розподілу	Типи завдань				
	1-й	2-й	3-й	4-й	загальний
FCFS	1,532055652	1,3368916	1,47227886	1,37791404	1,3779661
LIFO	1,5179405	1,365694	1,3799023	1,4071966	1,3662936
HPF	1,2818715	1,1678487	1,2491847	1,1249648	1,1720267
Backfill	1,024695	0,9477967	0,966164	1,055375	0,9642994
Backfill_mod	**0,9141202**	**0,9375502**	**0,969127**	**0,9475**	**0,9**
Simplex	1,015221	0,997148	1,060385	0,951856	0,97272
Smart	0,987972	0,956344	0,903608	1,070198	0,951149

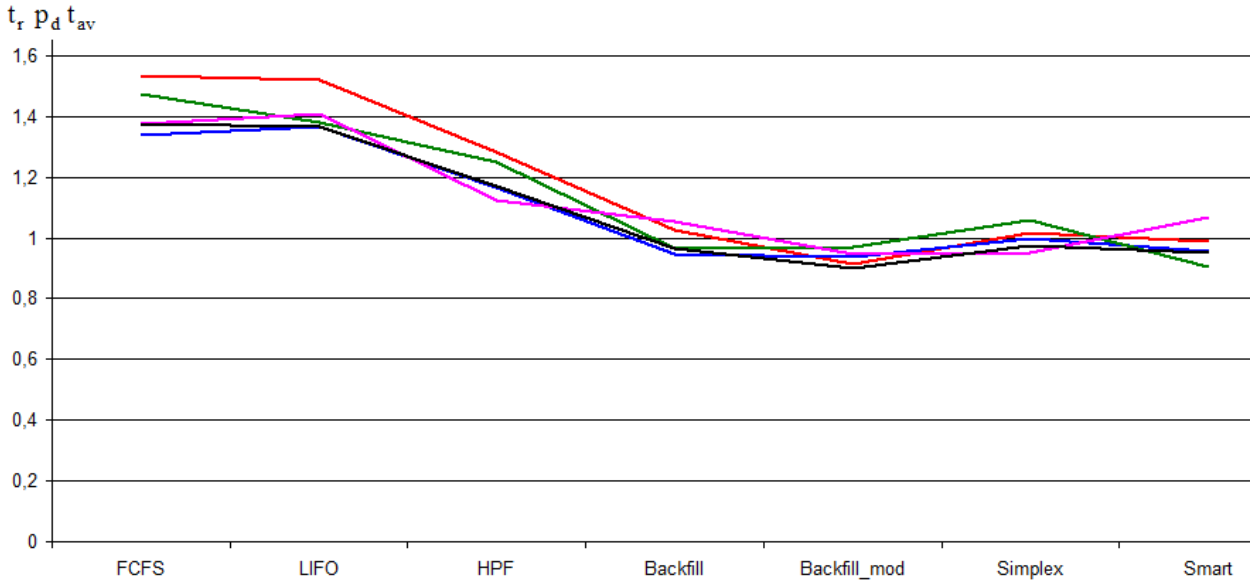

Рисунок 3.13 – Розподіл вхідного пулу завдань на обчислювальні ресурси кластеру відповідно критеріям t_r, t_{av} та p_d

В даний час не існує ідеального методу розподілу, який для будь-якого пулу завдань дозволив би отримати оптимальний варіант розподілу завдань на обчислювальні ресурси за обраними критеріями розподілу. Однак, якщо

користувачеві відома низка характеристик обчислювальних ресурсів відповідного кластеру, а також технічні характеристики вхідного пулу завдань, то можливо здійснити експериментальне моделювання використовуючи середовище GRASS та в подальшому обрати оптимальний метод розподілу для конкретного пулу вхідних завдань та обчислювальних ресурсів. Далі, обраний план розподілу можливо запропонувати планувальнику в якості найбільш ефективного рішення, що в свою чергу розширить його можливості, а також підвищить ефективність використання обчислювальних ресурсів кластеру і подібних систем.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було отримано наступні результати.

Проведено аналіз існуючих методів розподілу вхідного пулу завдань в кластерних системах, який виявив, що існуючі планувальники мають ряд недоліків, головний з яких – це орієнтація на конкретний клас задач.

Розглянуто існуючу математичну модель розподілу вхідного пулу завдань в кластерних системах. Використання складових математичної моделі розподілу дозволяє знаходити оптимальний план розподілу, але тільки за одним критерієм.

В ході проведення аналізу завдань, які надходять на вхід кластеру було виявлено що вони носять різнорідний характер. Тому при розподілі їх слід класифікувати відповідно за даними, якими вони оперують (вхідні та кінцеві дані).

Аналіз критеріїв оптимізації довів, що їх використання буде більш доцільним, якщо при розробці плану розподілу обирати ці критерії не по одинці, а у різноманітних варіаціях.

На основі даних, які були проаналізовано вище було модифіковано математичну модель розподілу вхідного пулу завдань у кластерній системі за рахунок введення до неї критеріїв розподілу, які допомагають обрати оптимальний план розподілу для відповідних пулів завдань та обчислювальних ресурсів кластеру.

На основі запропонованої математичній моделі було реалізовано модуль пошуку оптимального плану розподілу для вхідного пулу завдань кластерної системи та впроваджено його до імітаційного середовища GRASS.

В ході дослідження було проведено ряд експериментів з розподілу вхідного пулу завдань на обчислювальні ресурси кластеру для різних методів розподілу. Під час їх проведення отримана залежність результатів розподілу від класу завдань, які надходять на кластер. Результати доводять те що,

застосування одного методу розподілу завдань в планувальнику є менш ефективним ніж множини.

Також результати, отримані в ході експериментів, свідчать про скорочення часу виконання пулу завдань до 9%, часу знаходження завдань в черзі до 12% та зменшенні відсотку простою обчислювальних ресурсів до 15% для ряду методів розподілу, які впроваджено в імітаційному середовищі GRASS.

Використання середовища імітаційного моделювання GRASS та модернізованої моделі розподілу завдань дозволять сформувати плани розподілу конкретного пулу завдань по кожному із методів розподілу, і отримавши результати моделювання, можливо зробити пропозицію використання методу розподілу, який надає оптимальний (найкращий) результат, планувальнику.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шликов В.В., Данілова В.А. Високопродуктивні розподілені обчислювальні системи: Практикум: навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки», спеціалізації «Інформаційні технології в біології та медицині» КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2018. 108 с.

2. Косолапов А.А. Аналітичні моделі масового обслуговування в задачах проектування інформаційних систем: навчально-довідковий посібник. Д.: «LikePrint», ФОП Гечка Т.О., 2015. 186 с.

3. Волк М.А, Филимончук Т.В., Гридель Р.Н. Методы распределения ресурсов для GRID-систем. Збірник наукових праць ХУПС. Харков: ХУПС, 2009. №1(19). С.100-104.

4. Бульба С.С., Давидов В.В., Кучук Г.А. Метод розподілу ресурсів між композитними застосунками. Системи управління, навігації та зв'язку. Збірник наукових праць. Полтава: ПНТУ, 2018. Т.4(50). С. 99-104.

5. Кірік О.Є. Розподіл ресурсів у розподільчих системах з оптимальним перерозподілом навантаження постачальників продукту. Системні дослідження та інформаційні технології, 2013, №4. С. 38-51.

6. Кучук Н.Г., Зубрицький Г.М., Кучук Г.А. Метод розподілу ресурсів в комп'ютерних системах на інтегрованих програмних платформах. Системи обробки інформації, 2022. №1(168). С. 36-42.

7. Демчик В.В. Аналіз сучасних методів планування обчислень та балансування навантаження в розподілених комп'ютерних системах. THE SCIENTIFIC HERITAGE, 2021. №72-1(72). С. 30-39.

8. Гребенюк Д.С., Давидов В.В. Метод первинного виділення хмарних обчислювальних ресурсів на основі аналізу ієрархій. Системи управління, навігації та зв'язку. Збірник наукових праць. Полтава: ПНТУ, 2020. Т.3(61). С. 80-85.

9. Кисіль В.В., Драч І.В., Кисіль Т.М. Модель задачі складання та

оптимізації розкладу занять за умови задоволення об’єктивних та суб’єктивних вимог навчального закладу. Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки, 2019. Том 30 (69). Ч.1, №6. С. 65-70.

10. Orlov V., Demianchuk B., Klimenko V., Serhii Y., Tarasov O., Semenenko L. Efficiency criteria for digital adaptive systems for remote detection and recognition of dangerous objects. Journal of Scientific Papers «Social Development and Security», 2021. №11(5). P. 119-132.

11. Литвинов В.В., Стеценко І.В. Управління розподіленими ресурсами ГРІД-системи. Математичні машини і системи, 2012. № 2. С. 3-12.

12. Филимончук Т.В., Волк М.А. Разработка модифицированного метода обратного заполнения Backfill для консервативного резервирования. Системи обробки інформації. Харків: ХУПС, 2017. №1(147). С. 33-37.

13. Yesil S., Ozturk O. Scheduling for heterogeneous systems in accelerator-rich environments, The Journal of Supercomputing, 2022. №78. P. 200-221.

14. Toporkov V., Toporkova A., Tselishchev A., Yemelyanov D. Slot Selection Algorithms in Distributed Computing, The Journal of Supercomputing, 2014. Vol. 69(1). P. 53-60.

15. Волк М.А, Филимончук Т.В. Обобщенный критерий оценки задания для технологии планирования заданий в GRID, Информатика, математическое моделирование, экономика: Сборник научных статей по итогам третьей международной научно-практической конференции, 2013. Том 2. С. 172-176.

16. Pokusin N.V. Load Balancing in Distributed Heterogeneous Computing System with a Priori Inde购nite Input Stream Nature, Naukovedenie, 2013. №3, URL: http://naukovedenie.ru/PDF/82tvn313.pdf.

17. Zhao H., Sakellariou R. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems, 13th Heterogeneous Computing Workshop (HCW2004), 2004, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.3069&rep=rep1&type=pdf

18. Голубев И. А. Планирование задач в распределенных

вычислительных системах на основе метаданных: дис. … канд. техн. наук: 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей». Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина). Санкт-Петербург, 2014. 135 с.

19. A. Amar, R. Bolze, E. Boix. DIET 2.8 user's manual, 2011, URL: http://graal.ens-lyon.fr/~diet/download/doc/UsersManualDiet2.8.1.pdf.

20. Moab Workload Manager v.7.2.4 Administrator Guide. Adaptive Computing Enterprises, 2013. 1136 c.

21. Maui v.3.2 Administrator's Guide. Adaptive Computing Enterprises, 2011. 287 c.

22. Букатов А.А., Хачкинаев Г.М. Разработка системы управления параллельными заданиями в гетерогенной вычислительной среде, Труды первой Всероссийской научной конференции «Методы и средства обработки информации», 2003. С. 197-202

23. Кайдан М.В., Думич С.С., Максимюк Т.А., Бурачок Р.А., Готра Л.М. Розрахунок параметрів якості обслуговування у фотонних транспортних мережах, Вісник Національного університету «Львівська політехніка». Радіоелектроніка та телекомунікації, 2014. №796. С. 147-156.

24. Feoktistov A.G., Kostromin R.O. Development and application of subject-oriented multi-agent systems for distributed computing management. Izvestiya SFedU. Engineering Sciences. 2016. №11(184). P. 65-75

25. Filimonchuk T., Volk M., Ruban I., Tkachov V. Development of information technology of tasks distribution for grid-systems using the GRASS simulation environment, Eastern-European Journal of Enterprise Technologies. Information and controlling system, 2016. Vol.3/9 (81). P. 45-53.

26. Северілов П.В., Гула К.І. Моделі оптимального розподілу ресурсів у вертикально інтегрованій системі. Вісник Вінницького політехнічного інституту, 2008. №6. С. 41-46.

27. Batool K., Niazi M.A. Modeling the internet of things: a hybrid

modeling approach using complex networks and agent-based models. Complex Adaptive Systems Modeling, 2017. Vol. 5, №1. P. 4, URL: http://dx.doi.org/10.1186/s40294-017-0043-1.

28. Филимончук Т.В., Волк М.А., Казмина Д.Р., Ольшанская Т.И., Рисухин М.В. Модифицированная информационная технология распределения заданий на ресурсы для систем облачных вычислений. Сучасний стан наукових досліджень та технологій в промисловості. 2019, №1(7). С. 121-128.

29. Волк М.А, Филимончук М.А., Филимончук Т.В. Модуль распределения заданий в GRID-системах. Системи обробки інформації. Харків: ХУПС, 2012. №2(100). С.177-182.

30. Філімончук Т.В., Колтун Ю.М., Климова І.М., Корнієнко Д.Ю. Модель розподілу пулу завдань за обчислювальними ресурсами. Системи управління, навігації та зв'язку. Збірник наукових праць. Полтава: Національний університет «Полтавська політехніка імені Юрія Кондратюка», 2023. Випуск 4(66). С. 151-154.

31. Толубко В.Б., Кожухівський А.Д., Вишнівський В.В., Гайдур Г.І., Кожухівська О.А. Імітаційне моделювання систем масового обслуговування: навч. посіб. Київ: 2018. 175 с.

32. Жерновий Ю.В. Імітаційне моделювання систем масового обслуговування. Львів: Видавничий центр ЛНУ імені Івана Франка, 2007. 307 с.

33. Quetier B., Capello F. A survey of Grid research tools: simulators, emulators and real life platform. In: 17th IMACS World Congress. Scientific Computation. Applied Mathematics and Simulation. Paris, France, 2005, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.8816&rep=rep1&type=pdf.